AN INTERACTIVE RASTER GRAPHICS SYSTEM AND LANGUAGE FOR ARTISTS AND DESIGNERS

By

STEPHEN A.R. SCRIVENER

THESIS SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
TO THE COUNCIL FOR NATIONAL ACADEMIC AWARDS

SCHOOL OF MATHEMATICS
COMPUTING AND STATISTICS
LEICESTER POLYTECHNIC

FEBRUARY 1981

ABSTRACT

This thesis is concerned with the design of a computer graphics system for in particular artists, and in general surface designers (where surface design refers to the design of 2Dimensional surfaces, e.g. wallpaper, carpets and fabrics). It is argued that the artist tends to be tentative both in terms of plans of action and also the meaning of the image. This is reflected in changes of mind that have consequences on the artists plans and on the structure of the perceived image (i.e. an image perceived in one way may be seen differently later on).

On both counts, it is argued, conventional vector based computer graphics does not possess the desired flexibility. Raster graphics employing a 'bitmap' to represent the picture for display offers new potential and greater flexibility. In particular it permits a view of interactive graphics, described as "communicating interpretations" in which the user is seen as being involved in communicating features in a visual scene shared by the man and the machine to the machine. In so doing the user is able to operate on objects in the picture as and when they are perceived.

A language (GLIMPS) is described which not only permits the user to generate pictures but also includes facilities for extracting and operating on perceptual features of a picture. GLIMPS makes it possible for both "physical" (region) and "non-physical" (figure on ground) properties of the 'bitmap' to be handled.

In conclusion it is argued that the "communicating interpretations" view is more generally applicable in interactive computer graphics.

ACKNOWLEDGEMENTS

To people : Ernest Edmonds
Malcolm Hughes
Lyndon Thomas
Ted Allwood
Debbie Scrivener
and all others unnamed.

To machines : PDP 8
HONEYWELL 516
THE LEICESTER RASTER DISPLAY
BURROUGHS B6700

To publishers and others:
Dr. Bela Julesz
Thames and Hudson Ltd
Pergamon Press
Dover Publications Inc
Arno Press Inc
John Calder (Publishers) Ltd
The Arts Council of Great Britain
Crown Publishers Inc
Macdonald Futura Publishers Ltd
Phaidon Press Limited

MANY THANKS

TABLE OF CONTENTS

INDEX OF FIGURES

CHAPTER 1

INTRODUCTION

For some, the idea of a visual artist using a computer as an aid to producing art may seem paradoxical. What, they might ask, can the most logical of machines offer a branch of human behaviour which by tradition embodies much of the irrational? And yet, since the early sixties many artists have spoken up in support of the computer, extolling its possibilities and demonstrating its potential. For many of the pioneers of computer art it was the very fact that the computer can be programmed to perform logical sequences of actions which seemed to offer a key to understanding the irrational; a way of automating the creative process. For others the computer appeared as a mere tool, but a tool of possessing power unequalled by more familiar technology.

In the beginning it was necessary for most of the artists who saw some value in the computer to become more than simple users of the computer, in the sense that a driver is a user of a car. They were forced by the lack of facilities to become computer programmers. This meant, continuing the analogy, that in order to drive the car it was necessary to become a car mechanic. Fortunately, for most of us a knowledge of what goes on underneath the bonnet of our motor car is not a pre-requisite for getting from one place to another. Between us and the engine there is a set of controls which make it possible for us to be users of the machine with limited knowledge of how it works.

Much of the work of computer science is concerned with devising ways of providing easier access for the potential user to computing power by reducing the new knowledge which must be acquired by the user, whilst at the same time providing a tool flexible enough to serve his purposes. As more artists began to show an interest in the computer so the computer scientist sought to provide purpose built systems for them. To some this might appear whimsical on the part of the computer scientist but this is not the case. Modern man has a certain reverence for the artist, recognising in him an individualism and independence which is closely associated with creative behaviour.

In practise, producing a computer system for almost anyone has proven to be most difficult even when the tasks it is intended to serve display a high degree of similarity and the user population exhibit uniformity of behaviour. How much more difficult then the problem becomes when the system aims to meet the needs of something so diverse and idiosyncratic as Art.

Chapter 2 reviews a number of computer systems which were produced with the artist in mind and identifies two problems which are not resolved adequately in them. The first problem is that of making the system as accessible to the artist as possible. The second is that of making the system useful enough to serve the artists purpose. The former relates to the characteristics of the man-machine interface, the latter to the image generative and manipulative facilities which are employed via the interface. They determine respectively what is referred to as the *ease of use* and the *usefulness* of a system. In chapter 2 it is argued that existing systems do not resolve both problems simulatneously but place an emphasis on one over the other. Thus they tend to be either *easy to use* but not very *useful* or quite *useful* but difficult to use. One of the issues attacked in this thesis is how to provide a system which is both *easy to use* and *useful*.

The computer as a medium for visual expression is bound to present a degree of unfamiliarity since the artist must use a formal language to instruct the machine to perform tasks. When working with conventional visual media the use of language plays little part and we should expect this externalisation of objectives and actions to be at the very least unfamiliar. The term *ease of use* is used in this thesis in reference to the accessibility of a system in the face of this unfamiliarity.

The *usefulness* of a computer system is determined by its success in serving the purposes of its users. Therefore in order to design a computer system for the visual artist we must know sometning of his purposes. There are, however, many levels of artistic purpose which can be identified. For example, at one level the purpose of drawing a pencil across a sheet of paper might be to produce a line, which in turn could serve the higher level purpose of constructing a square.

Perhaps, at the highest level there is the artist's purpose in making art, but this is something which in general even the artist cannot explain with certainty and consequently we cannot know much about it. Also it is likely to be particular to the individual artist and since the objective is to design a system which can be useful to many artists then levels of purpose must be identified that have some generality.

The assumption that the purpose of the artist is to produce pictures provides the desired generality, and as a starting point is helpful because by excluding other forms of visual art (e.g. sculpture, construction, conceptual art) it reduces the problem area to manageable proportions. More significantly, however, it makes it possible to treat conventional picture making media as models for determining the design objectives of the computer graphics system which is the subject of this thesis.

Before discussing this point in more detail a slight detour is necessary in order to identify the prospective users of the system. An architectural plan is a model which helps the architect to visualise the building it represents. When the architect makes a change to the plan what has changed is his conceptualisation of the building which forces the modification to the drawing. The architect then, operates on an abstract building which is modelled in the plan. For the artist, however, the picture is the product. It is not so much a representation of the object of interest it *is* the object of interest and much of the manipulation of the picture takes place in response to its perceived pictorial properties. The significance of this is discussed in chapter 7 and it is only introduced here in order to observe that there are other design activities in which the function of the picture is similar to that of the fine artist. Graphic and textile design are examples for which this observation is valid. More generally, it holds true for all surface design. In the remainder of this text it is the artist who appears as the subject of the arguments, but they apply equally to the surface designer who consequently should be regarded as a potential user of the computer facilities which are described in the text.

In the ensuing chapters aspects of the picture making process are examined in order to provide answers to two important design questions. The first asks how the artist goes about making pictures; the second asks what kind of operations he performs on the picture during its production. The first question is concerned with the artists working process, the second with the picture generative and manipulative functions permitted by the medium. In the case of either question we should not expect the answer to be highly specific since the making of pictures is a personal and idiosyncratic business that provides as many answers as there are artists. However, general though the solutions may be, they provide important guidelines for the design of a computer system having sufficient flexibility to be personalised by the artist.

The question of how the artist makes pictures is examined in chapter 5 using examples of the working processes of a number of artists. It is argued that planning is a factor which plays an important part in characterising the artists interaction with his medium. The degree of planning undertaken in picture making is shown to differ when comparing artists and also to vary during the picture making activity of the individual artist. A computer system must permit various levels of planning to be implemented by the artist and also cope with changes in the plans during the interaction between the artist and the machine. As stated previously, the interaction will take place via the man-machine interface and it is this which must bear the brunt of the artists procedural demands, and in so doing allow the artist to work in his own particular way.

The question of what facilities should be provided for picture generation and manipulation is answered by looking at the conditions which cause these processes to occur (chapter 6). It is found that they can arise in response to what the artist perceives in the picture in progress. The mechanisms for manipulating picture elements in conventional computer graphics systems are described in chapter 7 where it is shown that they are inadequate for handling the manipulation of structures perceived in the screen, and then demonstrated that a raster scan graphic display can be used to overcome this problem. A raster scan display makes use of conventional television technology, and its name refers to the way in

which a picture is projected onto the T.V. screen in what is basically a series of horizontal, or raster, movements (i.e. 625 lines). When a picture is presented on a raster scan display by computer it must first be represented in computer memory, from which it is read out onto the T.V. screen. Currently a number of techniques are used for representing the picture in memory. The method adopted in the system described in this text is called a *bitmap*. The particular advantages of the *bitmap* technique are described in detail later (chapter 7). In brief, the entire screen, which appears as a matrix of points, is represented in the *bitmap* such that each screen point has its own memory location which records its physical state (e.g. intensity). Thus there is a one-to-one relationship between points in the *bitmap* and points on the screen. The essential novelty of the approach to manipulating pictures which unfolds in the latter stages of this thesis is the emphasis on processing the picture memory which is used to represent the picture for display on the television screen. Typically this processing involves extracting structures from the picture memory as and when they are perceived by the user and consequently removes the necessity for the user to pre-specify, before their manufacture, the pictorial structures to be manipulated (which he is often not in a position to do but is a pre-requisite in conventional systems). Picture memory processing (chapter 8) is a development arising out of this project which is regarded as having a relevance to interactive computer graphics as a whole (Scrivener et. al., 1978; Edmonds et. al., 1980; Scrivener, Edmonds, 1980).

In conclusion, although it has been stated that conventional picture making media provide models for the design of the computer graphics system discussed in this thesis, the system is not viewed as a model of a picture making medium. The argument for using a computer must surely be that it can offer facilities and possibilities different to those provided by conventional media. It is the extent to which the computer as a creative medium is successful in this respect that it may ultimately be seen to serve the artists Purpose.

CHAPTER 2

TWO APPROACHES TO THE PRODUCTION OF PICTURES BY COMPUTER

Since the early sixties, when the first generation of computer peripherals which permitted the input and output of graphical data appeared, there has been a proliferation of computer art. At first the production of computer art was practised by scientists having an amateur interest in art and a desire to use the computer creatively. More recently a number of professional artists have made use of the computer. In the following pages two approaches to the use of the computer as an aid in the production of pictures are examined. The first approach treats the computer as a sophisticated tool, whilst the second sees the computer as an assistant, or creative partner.

2.1 THE COMPUTER SCIENCE INFLUENCE

The first computer graphics to attract the attention of artists were produced by scientists, often without any explicit aesthetic objective. W. Fetter's work for Boeing Airway was concerned with the visualisation of a pilot in an aircraft cockpit, figure 1, and also airport flight simulation. The graphical data provided an aid to aircraft cockpit and airport design, by revealing the physical characteristics and limitations of man. Fetter was concerned with assisting the designer to deal with ergonomic, or human factors problems. However, the drawings which resulted, particularly those generated by the 'fifty percentile pilot project' proved of interest to artist's and have been exhibited in the context of art (Reichardt, 1968). In the first place they are aesthetically pleasing but more importantly they revealed to the artist that the computer might be used to assist the visualisation of three dimensional objects and also to simulate processes.

Other scientists who influenced computer art were K. Knowlton and B. Julesq, both of whom were working at Bell Labs.. Julesq (1965), a psychologist, produced computer generated pictures, figure 2, comprised of dots which he employed in experiments on texture descrimination in the human visual system. Knowlton has been involved with computer graphics and computer art from the outset.

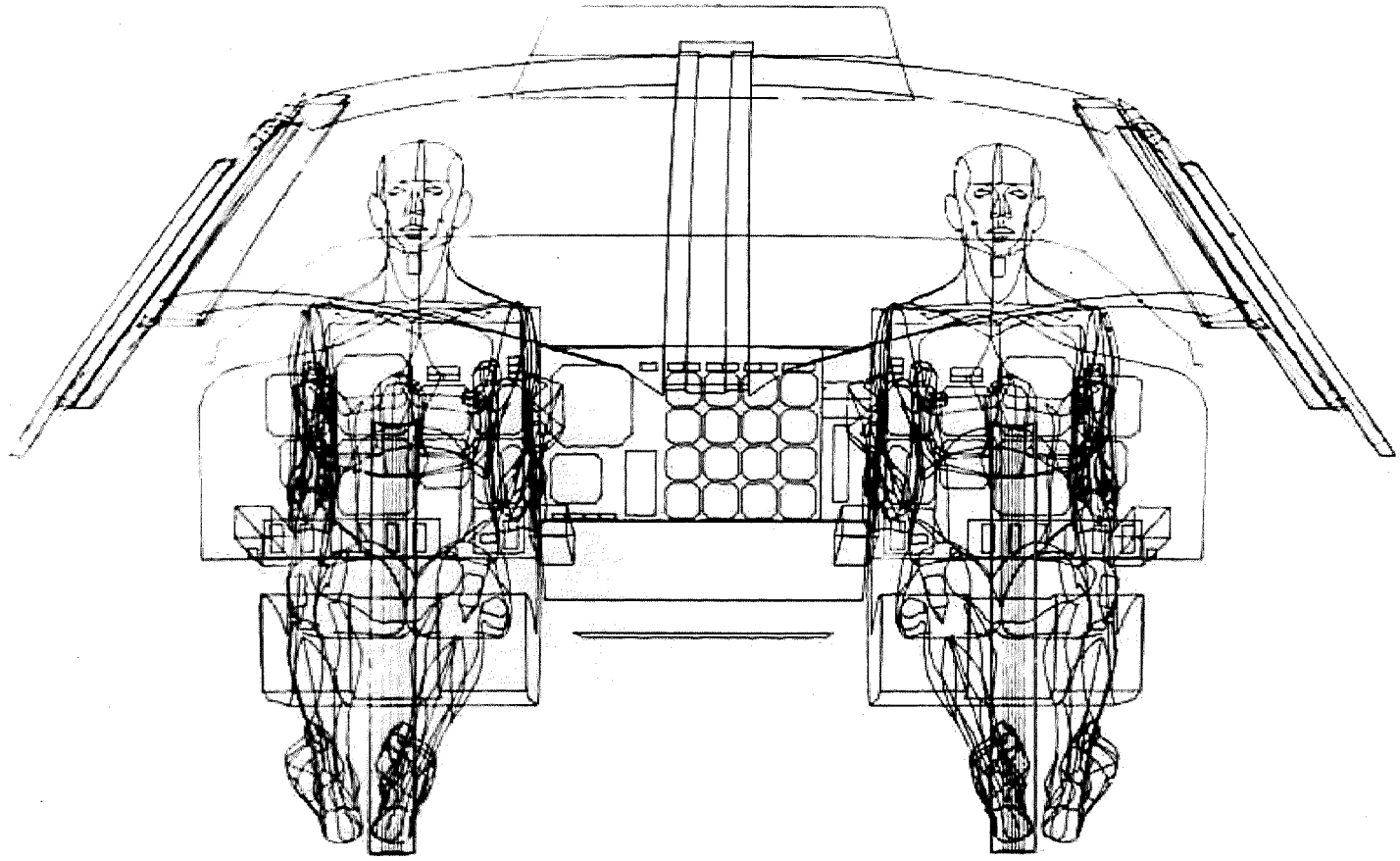


Figure 1

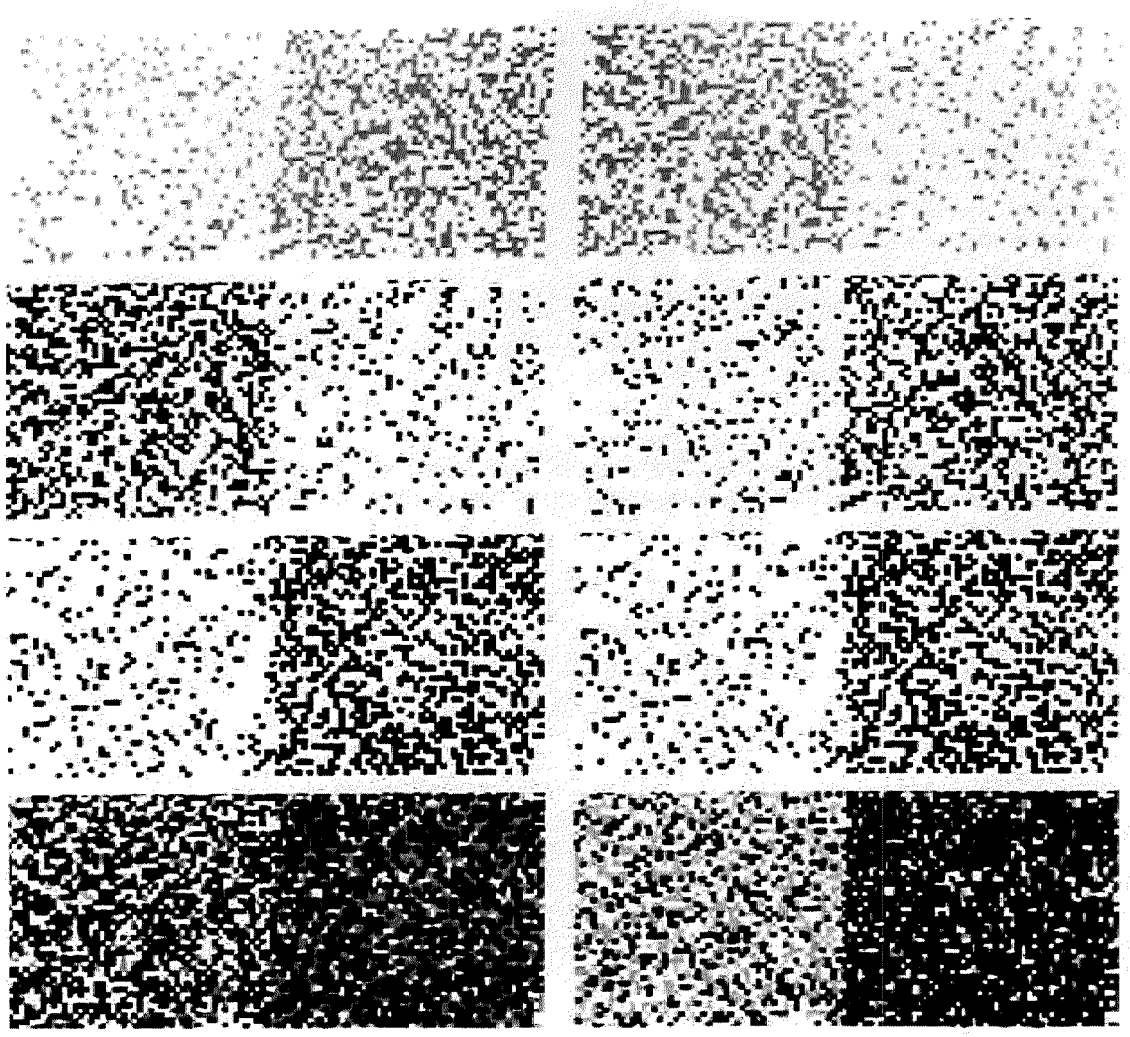


Figure 2

His most familiar work is the series of graph plotted pictures produced by the technique of 'picture processing' which uses a computer to transform a photograph into a digital form, figure 3. Whilst Fetter and Julesq do not make any aesthetic claims for their work, Knowlton (1969) has always been conscious of the artistic implications of his work. Indeed, he defined one of the objectives of his work as being "to explore new forms of computer produced art".

Whatever the personal or scientific objectives, the work of Fetter, Julesq and Knowlton was important because it brought the computer to the attention of the artist, and revealed some of its possible artistic applications.

2.2 THE POSSIBILITIES

As computer graphic facilities became more generally available, particularly in educational establishments where both computer science and art were studied, so artists began to make use of them. During the last decade the number of artists who use the computer has increased significantly and many publications and a number of books have appeared to chart their progress. Authors often predicted a significant transformation in art as a result of the artists interaction with the computer. The atmosphere of the period is expressed by Charles Csuri (1974) who wrote that, "Many of us became keenly interested in the alternatives to traditional ideas afforded by computer processes. These ideas and feelings soared into some kind of n-dimensional space with the expectation and anticipation that one was on the threshold of a new and revolutionary approach to art". Unfortunately this was not to be the case for as Csuri admits, "The nasty details of communication and implementation and the niave conceptions the artist had about the computer quickly brought frustration and rather modest results.....the artist may want a graphics programming language with an easy-to-use command syntax, but he is stuck with Fortran, not enough core, or an assembler language and a printer instead of a drum plotter or a refresh CRT".

Whilst there is still a need for better computer facilities for the artist the persistence of a few dedicated artists has resulted in work which suggests that the initial optimism was justified.

2.3 THE TRENDS

Today, two destinct approaches to using the computer in the production of pictorial art can be observed from an analysis of the work and writings of a number of artists. From the first view the computer is seen as a tool, rather like a sohpisticated paint brush. It functions to assist the artist to do what he already does using more conventional methods and is regarded mainly as a production tool. The second view presents the computer as a kind of assistant, even an autonomous creator or extention of the artist. From this viewpoint the computer is not seen merely as a production tool but as a decision making partner.

2.3.1 THE COMPUTER AS A TOOL

As in the case of most new tools, artists have tended to explore ideas carried over from previous mediums. The early use of the camera, for example, is characterised by its debt to painting, in the sense that photographs reflected the studied composition, subject matter and treatment of the paintings of the period. It might be described as painting with a camera.

Likewise, early computer art is characterised by the use of modules, or basic elements, which are transformed or combined using rules. Also random and probibilistic functions play an important role. These are features not unique to computer art and can be shown to characterise earlier work produced without the aid of the computer.

2.3.1.1 RANDOMNESS, RULES AND INTUITION

During the first quarter of the twentieth century the use of random processes became a feature in the production of much art. Jean Arp would drop pieces of string onto his canvases accepting the arbitrary arrangement that followed as the basis for linear outlines, figure 4. Duchamp, Ernst and many of the Dadaists and Surrealists explored the use of randomness in pictorial art. In poetry, similarly, ramdomness was introduced into the productive process by Andre Breton and others. The use of randomness in art was closely connected with the

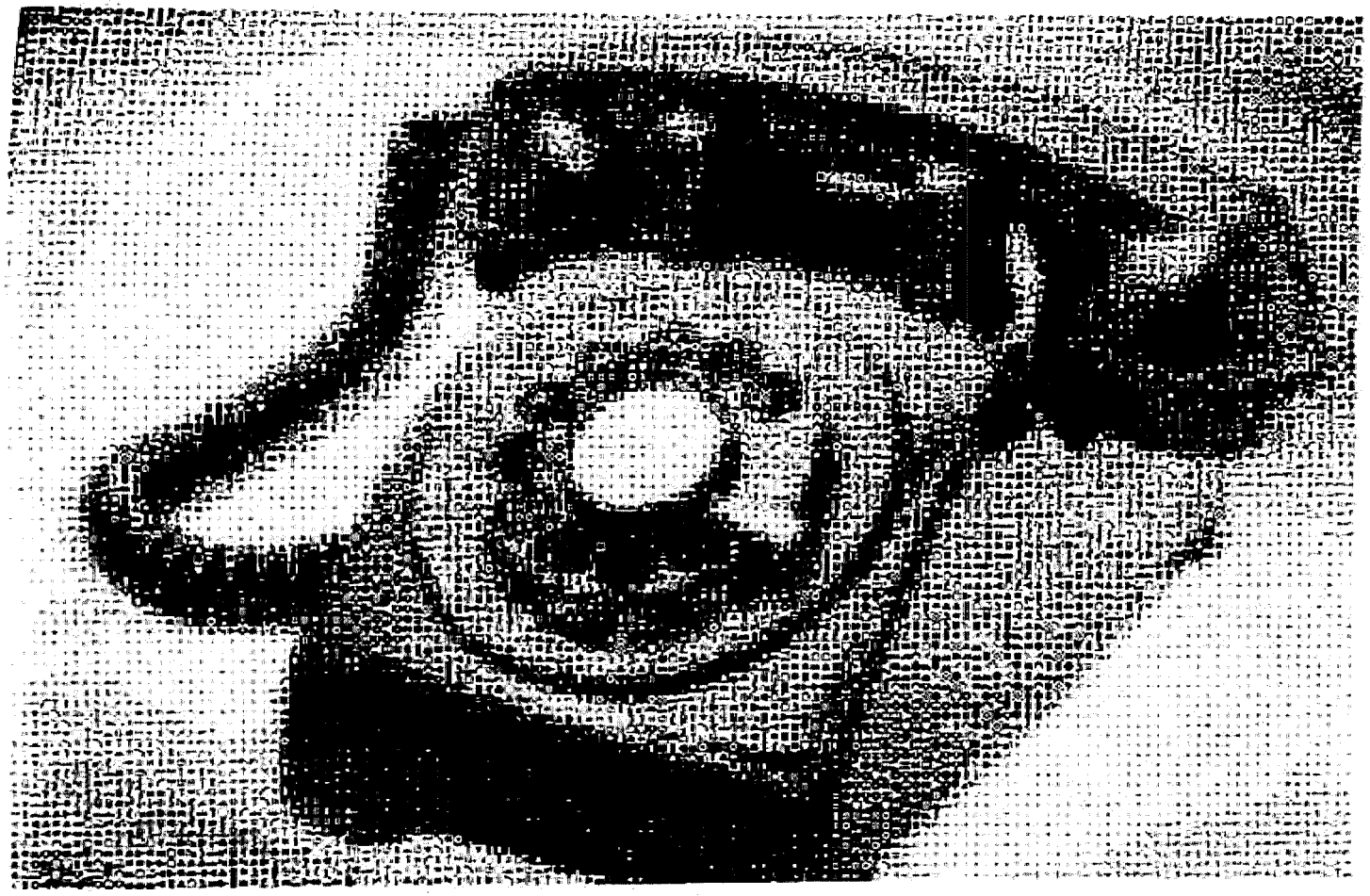


Figure 3

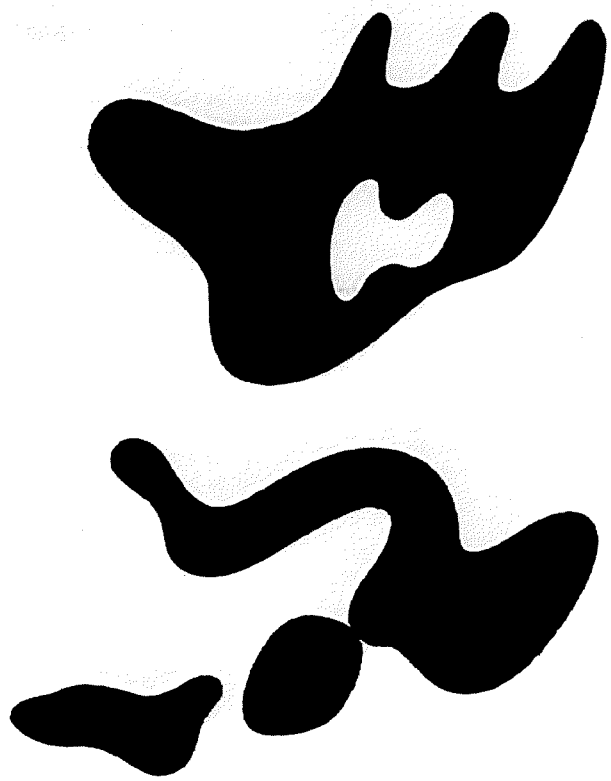


Figure 4

psychological theories of Freud, and more particularly Jung, which stressed the importance of the subconscious mind. Randomness was seen as one way of providing a stimulus more directly available to the subconscious mind.

The use of randomness in art has persisted although the reasons that prompt its use are different to those of the Surrealists. However, it can be argued that the function of random and chance processes remains the same. The artist is just as susceptible to convention as any other person. The young artist assimilates the rules for picture making as professed by his masters and as expressed in the work of previous artists. The word intuition is often used to describe the mechanism which permits the artist to make *creative leaps*. However, few artists make *creative leaps* and most must be content to consolidate the promise of other artists *creative leaps*. The more general manifestation of intuition is in the automatic application of picture making rules assimilated through learning and experience. Intuition applied in this way traps the artist in convention, circumvents reason and makes it difficult for him to break new ground.

Whilst the artist may not be able to predict the outcome of his actions precisely there may be a predictability to his behaviour which makes it inevitable; it is not easy for the artist to surprise himself. Leaving certain decisions to chance makes the outcome of the picture making process unpredictable, from the artists point of view.

Thus Arp might choose the length of his piece of string (line former), the height from which it is to be dropped and its position over the canvas. Its actual position and arrangement being determined independently of the artist as a result of the arbitrary physical conditions governing its descent to the canvas. The unpredictability of the outcome in turn creates the possibility of the unexpected in the picture; in other words the result may be a surprise to the artist. Consequently the use of random processes can be seen as a way of circumventing the constraints imposed by intuitive decision making processes; and surely functioned in the same manner for the Surrealist.

An alternative way of avoiding automatic decision making is to specify precise rules governing the generation of pictures which pre-determine

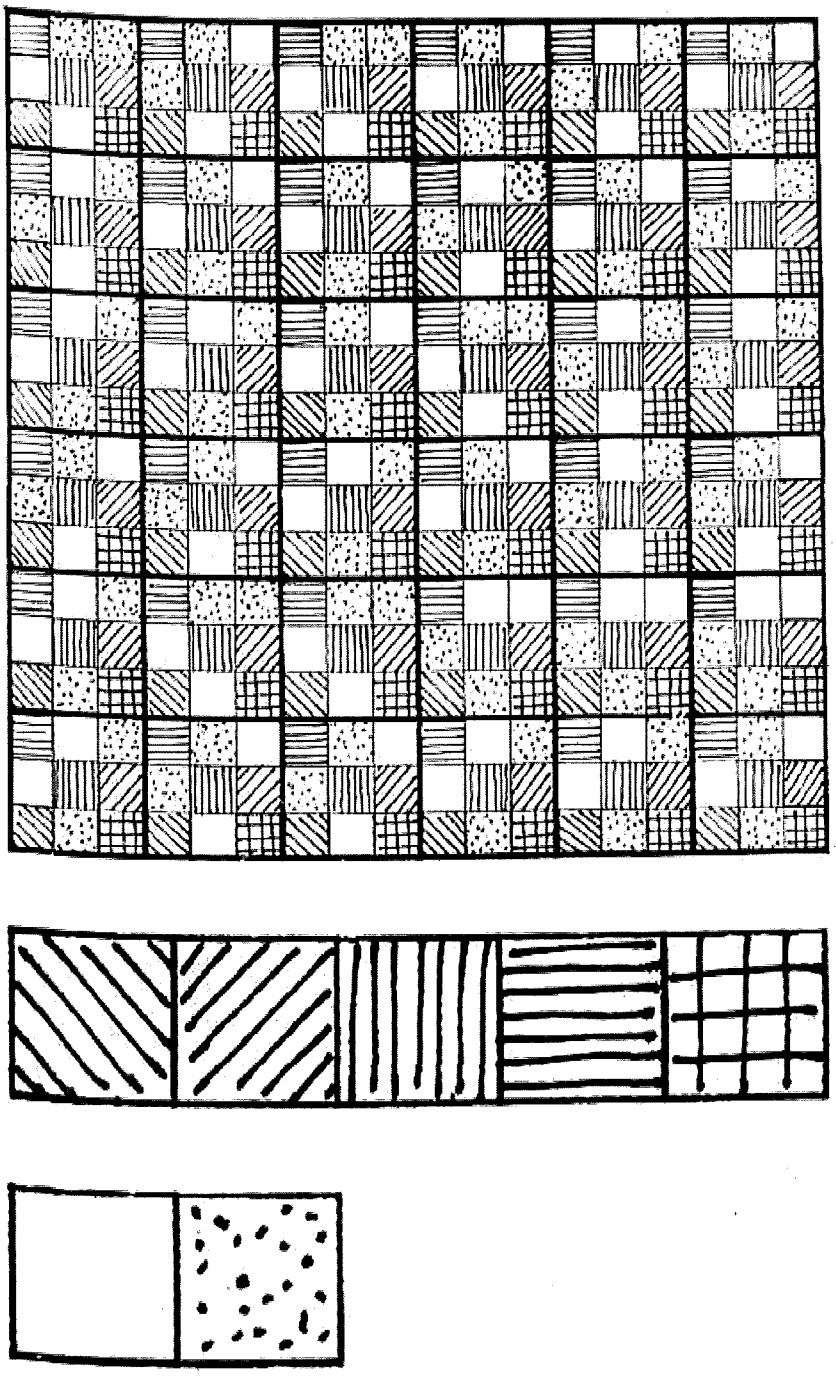


Figure 5

the decisions to be made. Whereas the use of random processes leaves some aspects of the picture to be determined by chance, the definition of precise rules leaves nothing to chance. Furthermore the properties of the resultant picture can be understood in terms of the rules used to generate it and consequently the artist can study and subsequently control its causes. Ideas about the use of rules in picture making have been explored, during the last two decades or more, by the Constructionists and System artists (System catalogue, 1972). Frequently rules are applied to the arrangement of basic pictorial elements or modules. Having defined the rules for the production of a picture, or set of pictures, the artist simply follows, without further decision making, his own programmes.

The search for tools to assist in the execution of random processes and rule governed programmes led some artists, almost inevitably, to use the computer.

2.3.1.2 SOME EXAMPLES

The Dutch artist Peter Struckyen (1973) has used the computer to assist in the production of works characterised by the use of modules, figure 5. Typically he uses the computer to establish the distribution of predetermined elements according to certain rules. For Struckyen. the computer serves as a tool for solving formal problems, such as establishing surfaces, spaces, limits, saturation and movement. The output from the computer is often transformed into a painting executed by the artist manually. FIXVAR, is a program which assists in the production of paintings which consist of 6 X 6 cells, each of which is divided into 3 X 3 positions. Some of the positions are FIXed by the artist; the remaining positions are VARiable. Given a FIXed start the program can complete an arrangement according to rules which aim to maximise on constraints, laid down by the artist, on the adjacency of elements. The completion of an arrangement is achieved by means of both rule governed and random processes. Typically the random processes take over when the rules fail to uniquely specify an element for a particular position in the matrix.

Sykora (1970) is another artist who has used the computer extensively, assisted by the mathematician J. Blazek. This artist often uses a

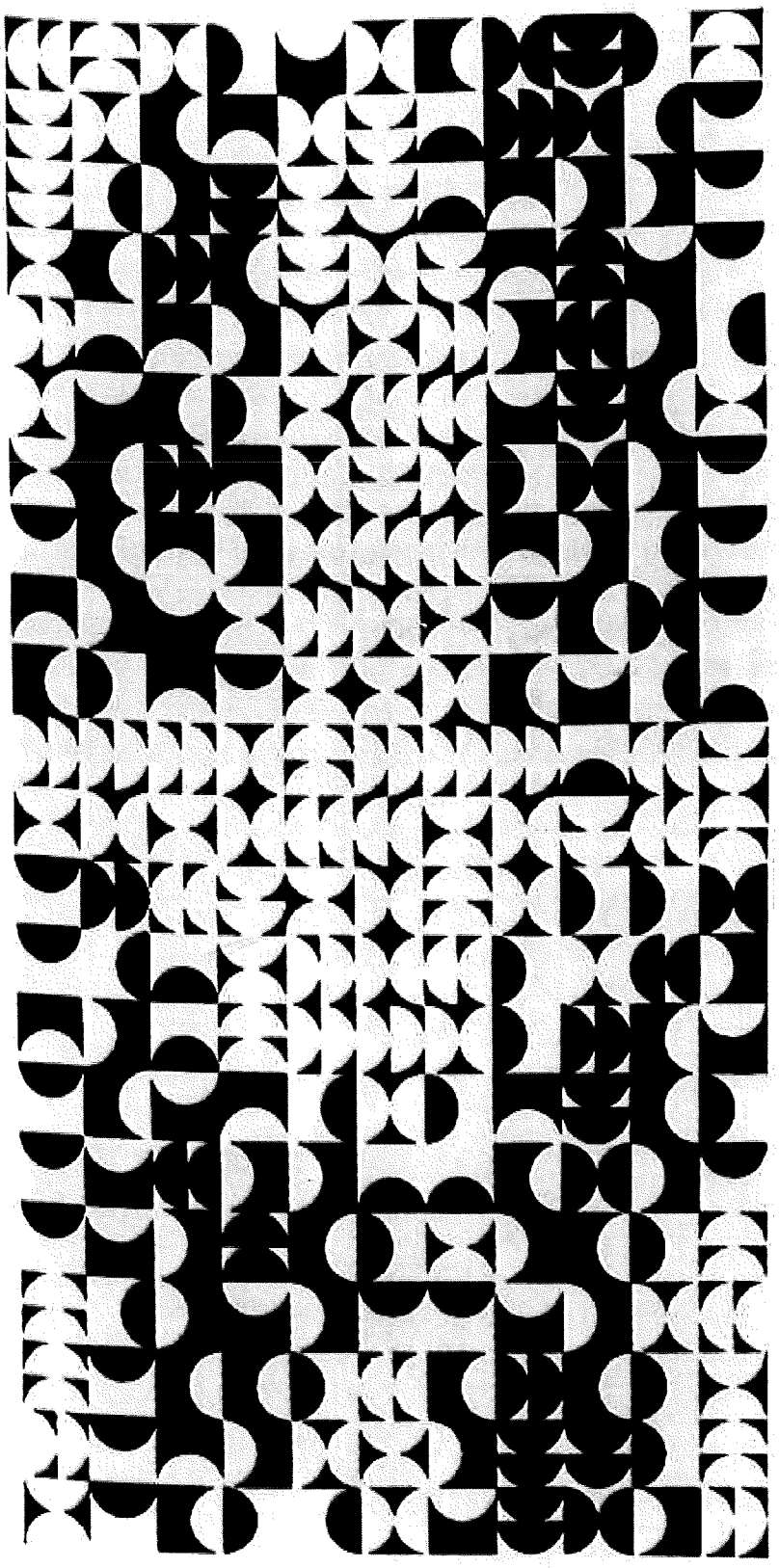


Figure 6

basic module with internal geometric pattern. One such painting, figure 6, resulted from the arrangement of elements on a surface of cells according to the rule that no two colours (the modules being divided so as to have different coloured edges) should be placed together. The artist has described the process of generating a picture by computer is as follows:

1. The elements are selected.
2. They are grouped according to colour density.
3. The artist then assigns a number of elements to the grid and also positive and negative values to signify where he wants the density of colour to increase or decrease.
4. One of four rules may be applied to the selection of elements to complete the picture.

The computer assigns elements in the matrix row by row, starting from the top left and returning along alternate rows from right to left until the matrix is completed. In the case where an element cannot be placed according to the rule selected the constraint is relaxed until one is found. In any case where more than one element could be assigned to a location in the matrix selection is decided by random process. Later the output from the computer is transferred onto a canvas.

This kind of computer art is indistinguishable from modular and system art produced without the aid of a computer. Indeed there is nothing about the ideas being explored which make the computer a necessary part of its production. Rather its use makes it easier for the artist to realise his ideas than existing methods.

2.3.2 THE COMPUTER AS ASSISTANT AND CREATOR

Whilst many artists continue to employ the computer as a tool, some artists have adopted a more radical approach which postulates a creative role for the computer. When an artist defines a program which performs functions he would normally perform he is, in a sense, automating part of his method. The act of programming can be seen as a process of externalising aspects of the artists creative method. The more aspects of method automated by the artist the more significant becomes the role of the computer, particularly if the artist automates

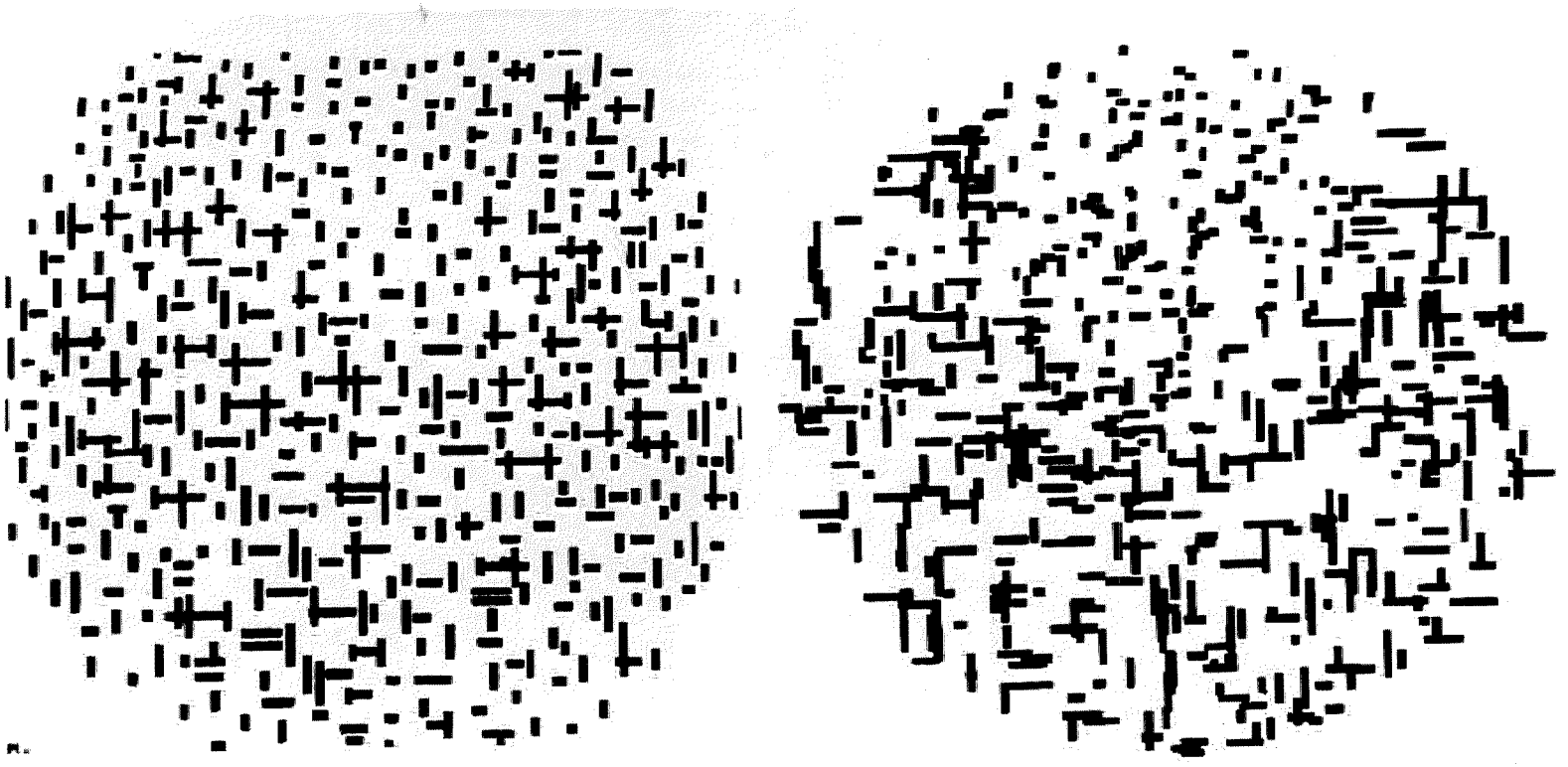


Figure 7

high level decision making processes. H.W. Franke (1971) writes that " ..as long as programs are restricted to those that merely capture routine processes of artistic creation then the trend towards theorising does not yet predominate. But with each move towards generalisation the question becomes even more insistent: how far can we go in programming the beautiful? Are there superior multi-level programs that incorporate general laws of aesthetics?"

2.3.2.1 MODELLING OF CREATIVE BEHAVIOUR

Ordinarily we judge whether something has intelligence by reference to some aspect of its behaviour. Turing proposed a test of a machines intelligence in which an interrogator proposes questions and tasks and attempts to descriminate between a man and a machine by their reponses (Turing, 1950).

A variation on Turing's experiment has been performed by A.M. Noll (1967) using Piet Mondrian's "Composition with Lines", figure 7. This experiment attempted to compare a computer produced picture generated according to a pseudo-random number generator with statistics chosen to approximate the bar density, lengths and widths in the Mondrian picture. Copies of the picture were presented side by side, to a hundred subjects. Results showed that 59% preferred the computer drawing and only 28% were able to descriminate correctly the Mondrian.

Noll argues that in terms of the Turing definition of creative intelligence this experiment shows that the computer has creative possibilities. The experiment also demonstrated that algorithms employing statistical and random processes can be made to manifest specific stylistic properties. Viewed another way an artist may be able to embody his style in the programs he writes.

2.3.2.2 SOME EXAMPLES

Generative Aesthetics, as defined by Max Bense (1971), is a mathematical aesthetics which distinguishes between the "material carrier" of a work of art and the "aesthetic state" achieved by the carrier. Generative aesthetics deals objectively with elements of the

"aesthetic state". These elements are pre-established and their appearance, distribution and formation are described mathematically. As defined by Bense "Generative Aesthetics therefore implies a combination of all operations, rules and theorems which can be used deliberately to produce aesthetic states when applied to material elements". Frieder Nake (1970) is perhaps the most important artist working in the area of mathematically based aesthetics using the computer. Nake attempts to define mathematical models which can be implemented on a computer for the production of aesthetically pleasing objects.

Micheal Thompson (1972), on the other hand, adopts a procedural approach. He has worked on software which simulates subjective decision making processes. One of his programs includes a "subjective visual model" capable of making "subjective" decisions when generating an arrangement of picture modules, figure 8. The model is based on an analysis of eye movements in the observer. Thompson uses networks to model eye movement representing the concepts of "tracking" and "skipping". The eye moves in short rapid movements looking at fixation points in a picture remaining motionless for 95% of the time. The term "tracking" is used to describe the illusory impression of the eye following features in a picture. "Skipping" is the term which describes the transference of the observers attention from one part of a picture to another.

Thompson uses specific modules (similar to those used by the spanish painter Barbadillo) which he represents by networks, figure 9. Numerical values are assigned to certain properties of the network, such as the length of an arc. When modules are placed together nodes may be connected thus creating new networks which may be evaluated. Thompson proposes that the scores provided by the networks represent the "tracking" of the picture. The "skipping" model is based on edge conditions. The scores from the two models represent the content of these subjective properties of a picture and the generative programs seek to produce arrangements with high scores.

Although the criteria used by Thompson for the design of the models are largely subjective his approach is unusual and sensitive to the fact that a large part of the artistic process is concerned with the

(b)
(a)

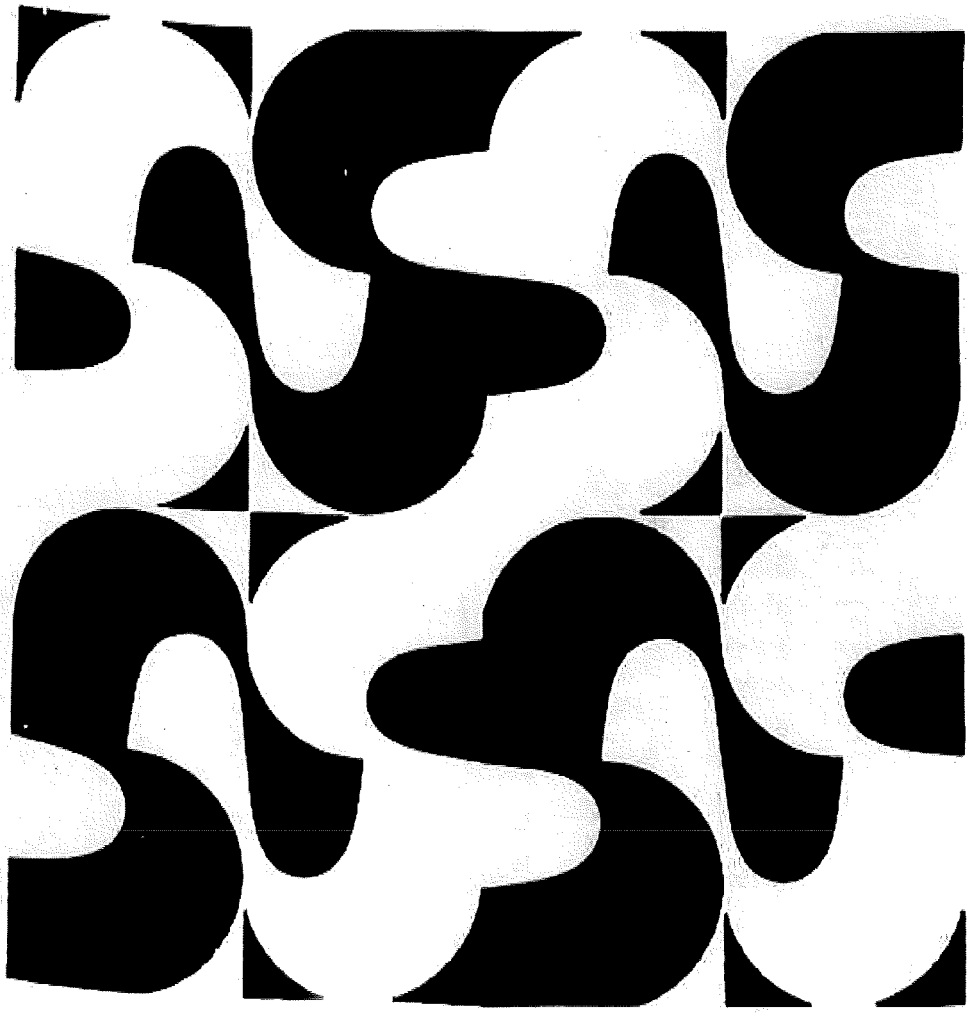


Figure 9

manipulation of such subjective experience. Furthermore an approach based on perceptual processes is more natural to the artist than mathematical and statistical methods. Thompson believes that the artist should use the computer not so much to produce art but to test and develop ideas on visual topics. However, to understand the visual he says, we must forget about geometry for "whilst most computer systems deal with geometric problems the artist is concerned with phenomenological problems - that is not what is there (geometry and engineering) but what we think we see". This is an important point (we shall return to a discussion of it in chapter 6) and is in sharp contrast to the work of Nake and Noll which rely on analysis of physical properties.

Harold Cohen (1974) is a painter who has recently become engaged in the use of the computer and adopts an approach that looks to what the artist does, and what determines his activity, rather than to properties of the object or perception, figure 10. Cohen believes that the computer can be useful to the artist if it can be demonstrated to offer something non-trivial, "if it can forward his purpose in some significant way". The problem he feels, is to "propose a structure which can be seen, as a whole to account for what the artist does. The notion of 'Purpose' might reasonably thought to characterise the structure as a whole", and that "the machine might be considered to advance the artists Purpose, if following the earlier argument, it could be seen that this might itself generate, or at least update, an appropriate notion of structure".

In describing the relationship between the computer and the artist he goes on to say "I identify the artist with the whole purpose-structure, the machine with the processes which are defined by the structures and in turn help to redefine it. Since under other circumstances these processes too would be played-out by the artist, I am also identifying playing-out with the computer to playing-out without the computer. For the machine to serve this purpose, the artist will need to use it as he uses himself".

This remark by Cohen allows an important distinction to be made. If the programs, produced by various artists were examined there would be little to distinguish them. It is only when the artists attitudes to

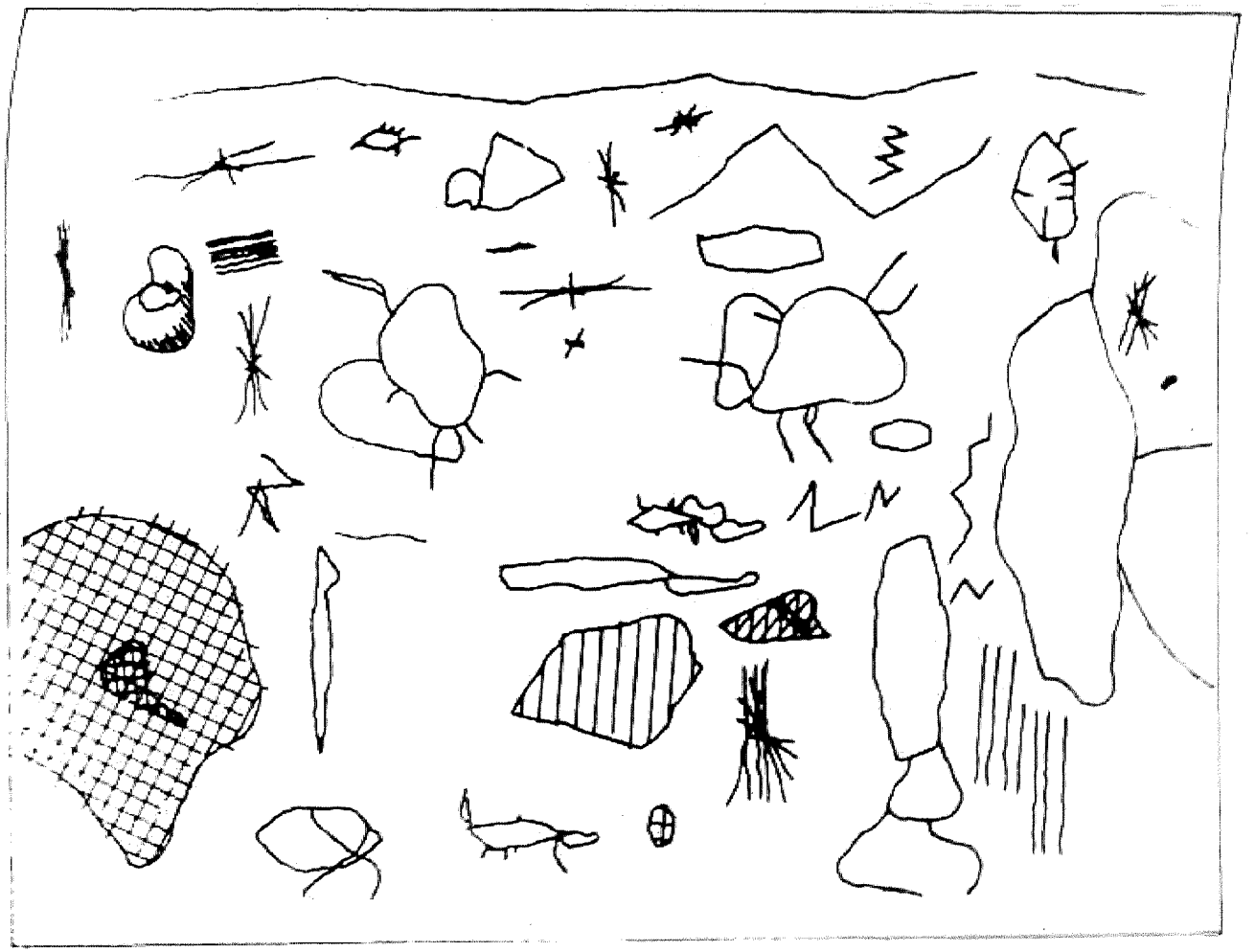


Figure 10

the use of the computer, and the way in which the processes they automate fit into their working process, that the differences become clear. The artist who regards the computer as a tool places a different value on its function to the artist, like Cohen, who regards it as an extension of the artist. Certainly a promising direction would seem to be one in which the computer is regarded as an integral part of the artists decision making process. What is important about this approach is that the computer is not regarded as just a tool used in a particular method but a machine which has knowledge of method ; which can take decisions and most important of all - modify method. It is in this way, perhaps that the *special* potential of the computer, which is for the most part only intuitively recognised, will become a reality.

CHAPTER 3

GRAPHIC SYSTEMS FOR ARTISTS

In general, the most interesting art produced with the aid of a computer has been done by artists who either write their own programs or collaborate with a scientist, as for example the Sykora, Blazek partnership. However, a number of systems have been produced specifically for artists. These systems tend to fall into one of two categories. The first category shall be called *easy to use*, the second *useful*. It will be shown that although one would expect a system to be both *easy to use* and *useful* these conditions are difficult to satisfy at the same time in a system.

3.1 *EASY TO USE* SYSTEMS

The design of the "*easy to use*" system is characterised by the desire to make the system as accessable as possible with the minimum of training.

3.1.1 ART-1

Professors Nash and Williams (1970), of the University of New Mexico, devised a program called ART-1. The program was intended as a teaching device to allow art students to produce simple computer graphics. ART-1 was implemented in batch mode, the results of a run being output onto a lineprinter. Tonal values could be modulated by the overprinting of user selected characters. It is possible to overprint two characters, this being achieved by the specification of the design in two arrays which are merged at the output stage.

The system consists of a number of subroutines which implement functions such as those for generating a line, a triangle or an ellipse. The artist controls the program by commands which are input to the system on cards in the form of numerical codes. Later Professor Williams (1971) extended the system, renaming it ART-2, to include improved shading facilities, figure 11. A number of statistical functions can be employed by the artist to control the distribution of the characters overprinted in a shape.

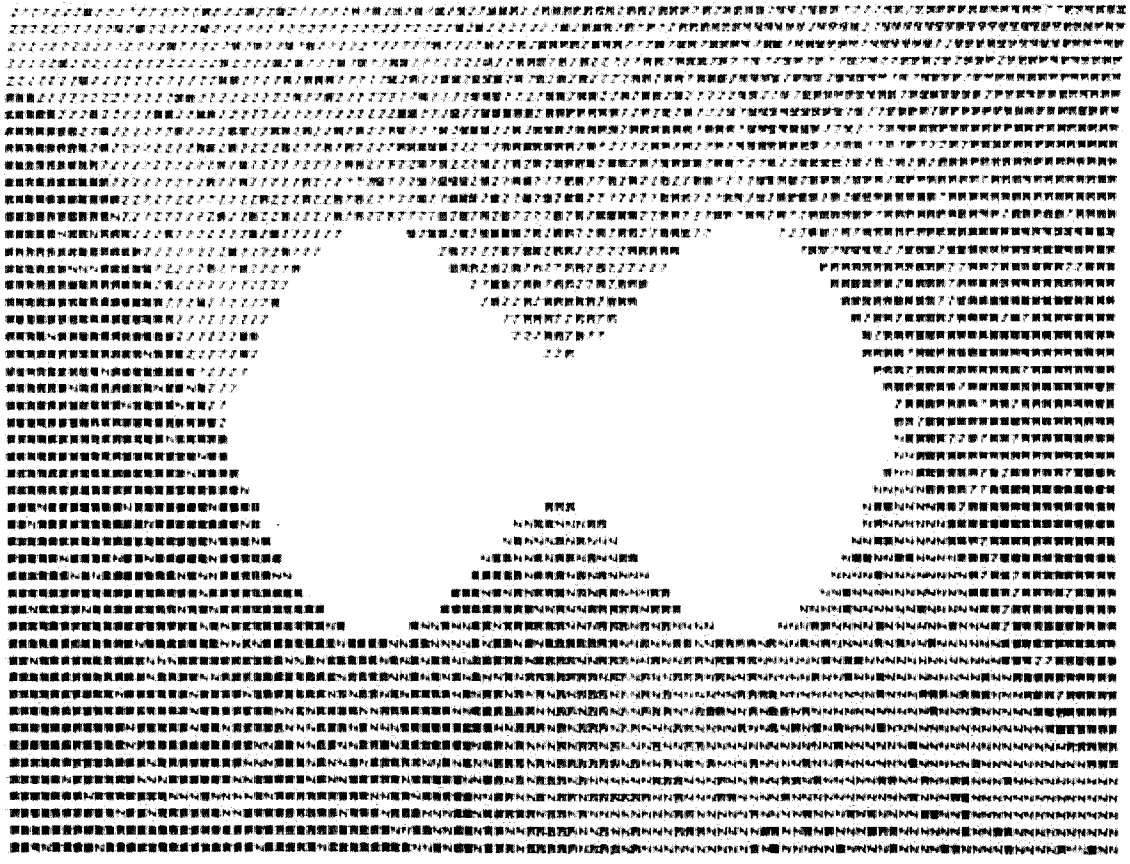


Figure 11

ART-1 is very restricted in the facilities it provides for the generation and manipulation of images. The artist is restricted to using a set of basic graphical forms (e.g. rectangle, circle, ellipse) with no functions which permit the run-time control of image generation and manipulation. In effect the set of cards the user inputs to the program represent an explicit geometrical description of the design.

Taking into consideration that the system was designed as a teaching aid it is clear that the authors felt that the less the art student needed to know about the computer and programming it, the better. This they attempted by the use of two techniques. The first simplifies command input to numerical codes which reduce the actual characters used in the design specification and thus the artist has to write less. Also complex shapes are provided as primitive commands which makes it possible to define a rectangle, for example, with one command rather than four line commands. In this way the number of commands required to define a design can be reduced.

Unfortunately, these techniques only reduce the input to the program, they do not guarantee *ease of use* or *usefulness*. The necessity of describing a picture in terms of numeric codes is both unnatural and difficult to master. There is, for example, no natural logic for associating a particular action (e.g. draw line) with a particular number. Furthermore, a users program is difficult to debug because the logic of the structure is not obvious from the language (numeric codes) used to define it. The use of complex primitives also presents problems. An analysis of the works of various artists would not reveal a repertoire of shapes common to them all. The emphasis on the generation of designs from a repertoire of shapes makes it difficult for an artist to use ART-1 to generate and manipulate more personal structures.

3.1.2 PLAD - PROGAMMING LANGUAGE FOR ART AND DESIGN

PLAD, developed by R. Saunders (1973) is essentially an extension of ART-1. It uses the basic functions of ART-1 whilst improving the command language. Implicit in the design of PLAD is the assumption that artists are unwilling to learn standard programming languages. Whilst this may be generally true it should be pointed out that much

computer art has been generated by artists proficient in one, or more programming languages. This is not intended to imply that standard programming languages are appropriate for the artist but it does indicate that he is capable of handling a high level of programming difficulty.

When considering the criteria for the design of a system for artists, Saunders observes that the usual means of specifying a system by reference to a "here is a computer, what do we want it to do" survey is inappropriate. The reason, he believes, results from the artist's attitude to new media, which is not to see it in terms of what he already does (i.e. I do this and the computer could do it just as well) but as something to be explored and examined; its possibilities emerging from this examination. This argument suggests that from the outset the artist must be able to explore computing. Saunders proposes an adaptive system responding to feedback from the artist and being extendable and open to change.

PLAD is divided into two parts. OPTION, the first part, allows the specification of output device and frame size. The second part, DESIGN, accepts sentences, consisting of a noun phrase and any number of verb phrases, defining graphic components and their arrangement. "The type of sentence", writes Saunders, "that can be used is very much like those that would be spoken over a telephone to another person who is reproducing the design by following instructions". The interpreter ignores redundancies, and recognised words used in the wrong context and assigns default values in the event of incomplete sentences. Saunders has also provided facilities for extending the command language and the functions it supports. In general, this would not be under user control, although in theory if a user can write FORTRAN programs then he could extend the system.

The command language of the PLAD system is an improvement on the ART-1 system in a number of important ways. It approximates more closely to english written text and consequently is easier to remember and is also self-documenting. The redundancies in the language however disguise an inflexible command syntax and the length of the input strings increases the likelihood of input errors.

More seriously, the command processor always attempts to complete, or modify a sentence in order to generate a syntactically correct sentence. This kind of facility has to be used with the greatest of care since the command processor may produce a set of actions different to those intended by the user when he specified the program. Thus although the program may run the results will be different to those expected. In this case, the user finds himself with a program which works but does not do what he expected it to do. Debugging the program, therefore becomes very difficult because the results do not accurately reflect the source program.

3.2 *USEFUL* SYSTEMS

"Useful" systems are characterised by the flexibility of image generative and manipulative facilities provided in comparison to the rather limited capabilities of the "easy to use" systems described in section 3.1.

3.2.1 SPARTA

Leslie Mezie (1969) has defined the design considerations of SPARTA as:

1. A data format which does not necessitate a mathematical representation of pictures, so that arbitrary line drawings of any complexity may be handled.
2. Inclusion of features normally available to graphic designers, e.g. line thickness.
3. Convenience to users.
4. Ease of use and learning.
5. Ease of addition.

The language has been implemented as a package of FORTRAN subroutines, and programming involves the specification of a series of CALL statements.

The data structure representing a drawing has various levels with a "point" at the lowest level, proceeding upwards through "curves" and "pictures". A "curve" is a collection of "points" which define a line and a "picture" is a collection of "curves". A "drawing", which only exists in graphic form when output on a graphical device, is

composed of one or more "pictures" all contained within the same "frame".

Pictures can be input explicitly by using cards or generated by program calls which include standard shapes such as circle, ellipse and spiral. The basic manipulation routines include those for moving a picture, altering its size, rotation and mirror image. A transformation routine allows any subroutine written by a user to be applied to the rectangular, or polar co-ordinates of each point of the picture. Facilities for the random distribution and distortion of pictures are also provided. Mezie does not elaborate on how his system meets his design criteria. Presumably he believes that they are met implicitly by the use of a FORTRAN callable subroutine package.

3.2.2 PICASO

The PICASO system produced by J. Vince (1979) of Middlesex Polytechnic is essentially a package of FORTRAN callable subroutines and functions. The package is large, consisting of over 100 subroutines and functions categorised in terms of control of the system, I/O, plotting, object and shape generation, shape manipulation, shape analysis, special effects and general functions. The system is not *easy to use* in the sense that learning the purpose of the various functions requires substantial effort and also a basic understanding of FORTRAN.

3.2.3 EXPLOR

Another FORTRAN based package is the EXPLOR language (Knowlton, 1975). The language is intended for generating two-dimensional patterns, designs and pictures from EXPlicitly provided two-dimensional Patterns, Local operations and Randomness. Designed by Ken Knowlton, of Bell Telephone Laboratories, it would seem to be a natural development of his earlier efforts in picture processing.

The output is to a line printer and up to three overprints are possible producing a four level grayscale. From the programmers point of view the system consists of a number of FORTRAN callable functions and subroutines. To use these effectively the user needs to learn a good deal of FORTRAN including subroutine CALL, GOTO, assignment, DO loop, logical IF statements, and the use of arrays. Knowledge of these

statements gives the user more scope for defining processes than SPARTA although, of course, more learning is required.

The user is asked to imagine the internally stored picture as a 140 X 140 array of picture cells, each holding a digit 0, 1, 2, or 3. At the beginning all cells have a value of zero and the language allows the contents of the cells to be manipulated in a variety of ways. Knowlton writes of the system, "Scientific and artistic applications include the production of stimuli for visual experiments, the depiction of visual "phosphenes" such as moving checkerboards and stripes, and picture processing. The system may be used to simulate a variety of two-dimensional processes and mechanisms, such as crystal growth and etching, neural (e.g. retinal) nets, random walk, diffusion, and interactive arrays of logic modules".

3.3 THE CONFLICT BETWEEN THE *EASY TO USE* AND *USEFUL*

SPARTA, PICASO and EXPLOR are more powerful than ART-1 and PLAD and have gained some popularity. In each case the increase in power is gained at the expense of accessibility. The designers have compromised *ease of use* for *usefulness*. The systems make it possible for the user to generate and manipulate more complex visual ideas but are not directly accessible.

On the other hand, the designers of PLAD and ART-1 attempt to provide a system which is accessible but by doing so conceed *usefulness* for *ease of use*. All the systems discussed so far fail to produce a balance between *ease of use* and *usefulness*.

The *ease of use* approach adopted by the designers of ART-1 and PLAD aims to simplify the man-machine interface. This is especially important if the facility is intended for use by artists who have no previous experience of the computer and no idea of its possibilities. In other words, if you hope to be able to say "here is a tool called a computer art system for you to use, now sit down and use it". In this context getting the man-machine interface right is pertinent, since a tool that is too difficult to use will probably be substituted for one which is tried and tested, and the system will fall into disuse.

ART-1 and PLAD fail to provide *useful* systems for the artist because in trying to make the system easy to use the designers restrict the artist to doing simple things. Although an artists approach to a new tool may be tentative at the outset, once its possibilities have been recognised the tool will be used with purpose and must be flexible enough to serve that purpose. ART-1 and PLAD are certainly more accessable than systems like PICASO, SPARTA and EXPLOR but they are so restricted in what they allow the artist to do that their "ease of use" serves no practical purpose. PICASO, SPARTA and EXPLOR on the other hand, require a level of understanding of the computer which makes them impossible to explore in the sense identified by Saunders. A weakness of all the systems discussed, with the exception of EXPLOR, is their emphasis on the provision of functions which effectively constitute a basic repertoire of shapes, such as a rectangle, a circle, or a horse in PICASO. Although this may help to reduce the size of the users program by providing language primitives which are complex visual structures, there is no set of complex visual forms which can be regarded as an appropriate description of the vocabulary of art. Thus the selection of any set to be included in a system is bound to be arbitrary. The emphasis should be less on the definition of functions which generate specific shapes and more on the provision of facilities for image generation and manipulation, definition of processes and overall control of the picture making process. The individual artist will know best what repertoire of shapes he wishes to generate and manipulate, if any.

One of the problems which must be resolved in the design of a system for artists is the interelationship between a systems *ease of use* and its *usefulness*. The structure of a system should be such that its "ease of use" does not over restrict its "usefulness" and visa-versa.

3.4 INTERACTIVE SYSTEMS

A limitation of the systems discussed so far is that they are all implemented in BATCH mode. Harold Cohen (1974) writes of the art making process that, "we associate with it an elaborate feedback system between the work and artist; and dependent upon this system are equally elaborate decision-making procedures for determining subsequent 'moves' in the work". Batch systems increase the time between 'moves' to such a degree that the process of feedback is severely retarded.

When the artist uses a batch system the tendency is for many moves to be pre-defined at each run. Consequently such systems are only useful when the artist has a more or less clear idea of what his objective is and how it can be obtained.

Although interactive computing offers a more natural approach at present few systems exist. Charles Csuri (1974), of the University of Ohio, has become increasingly involved in the development of interactive systems. One such system can be used for producing films. Drawings stored on disk are used as data for an animation program. A light pen is used to describe the path of movement of the object displayed on a CRT. Several paths and objects can be described at the same time. Parameters controlling speed, rotation and transformation changes for each drawing on the screen are typed in by the user. The movement of the objects along their various paths are then displayed on the CRT where they appear animated. Many possibilities can be tried out in a short period of time.

Another of his systems can be used to produce sculpture. Two basic procedures are involved; one for generating x, y, z co-ordinates which represent a 3-dimensional form, the other analysing the form for a continuous path, 3 axis milling machine.

A unit cube is used to develop a 3-dimensional surface. The artist has the option to define the boundary curves on each of 4 faces of the cube. Once established the boundary curves are approximated by equations. Alternatively boundary curves may be pre-defined and read in as data to the program. Plotted views of the surface, with hidden line removal, and as seen from any viewpoint may be produced. When the artist is happy with the design the co-ordinates representing the object are passed to programs that control milling machines and the final sculpture produced.

All this takes place interactively and although the forms which can be manipulated are limited the system does assist the artist to make decisions about a sculptural form by providing representations which allow the artist to *experience* aspects of the sculptural form prior to its manufacture.

Interactive versions of SPARTA and PICASO have also been produced. Known respectively as ARTA (Mezei, Zivian, 1972) and the PICASO INTERPRETER (Bradly, 1979) they are a considerable improvement on the original batch versions. ARTA is a conversational program that allows the user to select functions from menu lists displayed on a visual display unit using a lightpen. If a selected function requires arguments these are input by means of a keyboard. The user can also use the lightpen to input graphical data. The PICASO INTERPRETER is the implementation of a sub-set of the PICASO functions. Commands are entered by the user via a keyboard and interpreted immediately. Like ARTA, the picture under construction can be seen developing on a visual display unit. ARTA and the PICASO INTERPRETER suggest that much of the *ease of use* problem can be resolved by means of interactive techniques.

3.5 CONCLUSION

The main points which come out of this discussion may be summarised in the following observations which provide a preliminary set of guidelines for the design of a computer graphics system for artists.

1. Interactive systems are more appropriate than batch systems because they reflect more closely the interactive nature of the picture making process.
2. Essentially there are two problems. The first is concerned with what facilities a system for artists should provide, the second is concerned with the interface that makes it possible for the artist to make use of these facilities. The *ease of use* systems discussed in this chapter concentrate on the interface problem but do not consider what the artist does, or how he does it. The *useful* systems on the other hand concentrate on the provision of facilities and more-or-less ignore the interface. This imbalance is destructive in both cases. The *ease of use* systems are very restricted in what they permit the artist to do and consequently once a certain level of understanding has been achieved fall into disuse. The *useful* systems are so difficult to use that they tend to restrict their user population to those artists who are already committed to using the computer; it is difficult to approach them with

an attitude of exploration. Consequently for a system to be attractive to a wider range of artists the two problems must be solved simultaneously. The system must be accessible and at the same time useful.

It can now be seen that the two interdependent problems, outlined in the introduction, of the man-machine interface and the facilities for picture generation and manipulation are central to this thesis. With respect to the former, however, the computer can be programmed such that its behaviour appears to display human characteristics which might lead us to a redefinition of the interface as, for example, *man-robot*. In the next chapter, before proceeding to the main issues of the thesis, a number of possible characterisations of the computer are described and the reasons for the particular approach adopted by the author discussed.

CHAPTER 4

THE CHARACTER OF THE SYSTEM

As stated previously the questions involved in deciding what kind of computer system to design relate, essentially, to two problems. The first problem is to decide what facilities should be provided and the second to decide how the facilities will be made available to the artist. The computer is different to other tools in that in order to use it we have to "talk" to it. This means that any facilities provided by a computer will be employed by the user via a non-trivial communication link, rather like asking a person to perform a task by instructions given over the telephone. In this sense, it is the mechanism at the computers end of the communication link that controls the tools provided by the computer system. Consequently, the particular *character* this mechanism presents will influence the kind of interaction that takes place. The *character* the computer system presents to the user can be of many kinds; servant, partner or mentor.

4.1 THE SYSTEM AS SERVANT

Gaines and Facey (1977) write, "It is helpful to envisage the computer system in the role of a (Dickensian) servant, courteously guiding and helpful, quick to answer requests for information, often apparently more alert than the master, and yet ultimately knowing its place". This view of the system puts the user in control of the events which take place, the computer system does what it is told to do, nothing more. This is in sharp contrast to Negroponte (1977), who argues that servile computers are "compliant" thus allowing the user to perform a series of actions which might have been done better some other way, never suggesting or trying other alternatives. In other words, a compliant computer makes no effort to see the problem in different ways. He illustrates this point with the following example.

> "Two trains are a hundred miles apart, separated by a stretch of track. They start moving toward each other at twenty miles per hour. At the same time, a bird perched on one of the trains for some unknown reason starts flying toward the other, at thirty miles per hour. Upon reaching the advancing train,

it turns around and flies back to the first, whereupon it reverses its direction, back and forth and so on. The question is: how much distance did the bird cover, flying back and forth, until the trains meet?"

"A compliant computer", Negroponte writes, "will grind out the sum of the series, and yet worse, probably will not interact with the user in any way to expedite the sum. A more creative solution to the problem is to take it out of the context of space and put it into time. Obviously, or not so obviously, the trains required two and a half hours to meet. We see at once that the bird must also have flown for two and a half hours and hence covered a distance of seventy-five miles". Negroponte is suggesting that the computer should not be so singleminded in obeying the user. Instead it should be clever enough to view the problem in different ways and advise the user of better solutions, or at least guide him towards them.

4.1.1 ALLOCATION OF FUNCTION

Another of Negroponte's criticisms of the slave paradigm is concerned with what he calls "partitioning" and what Eason (1979) calls "allocation of function". The table below is taken from Eason's paper and shows the typical allocation of functions between man and computer based on assumptions about excellence of performance.

Computer Good	Man Good
(Man Poor)	(Computer Poor)
Speed of Processing	Pattern Recognition
Accuracy of Processing	Goal Formulation
Large Scale Memory	Resolving Ambiguity
Rapid Transmission	Recognising Novelty
	Creativity
(Pre-programmed)	(Self-programmed)

The fact that the man and machine appear to have complementary characteristics has led some authors to speak enthusiastically of the potential of man-computer symbiosis. The idea being that the computer can handle the routine tasks leaving the man free to concentrate on the creative aspects of the problem. Eason uses the

table simply to make the point that the computer, despite the efforts of workers in artificial intelligence, remains relatively non-adaptive in contrast to man. He argues that the last 10 years of research has not led to a liberation of man's potential and "leaves us with task performing systems that are no more adaptable than systems comprising man and less sophisticated technology" (Eason, 1979).

Negroponte argues that the fault is due to "partitioning" which, he writes, causes a lack of redundancy of tasks such that, "When each party is doing that and only that in which he, or she, or it, is expert; a premature sense of completeness arises, and a premature critical judgement is involved" (Negroponte, 1977). Eason believes that in order to explain the lack of success in man-computer interaction we must examine task characteristics. He identifies two task characteristics of structure and frequency. The implications of task frequency will be discussed in chapter 9 but task structure is of interest here. At the extremes of task structure are closed and open tasks. Eason characterises design aid and problem solving systems as having open task structure. Open ended tasks are best defined, he writes, "...by the characteristic that it is difficult to predict the exact nature of the transformations to be effected". A task is considered by Eason, to involve the transformation of inputs to outputs which meet goal requirements. From the standpoint of the system designers such tasks pose a very difficult problem because he has to predefine a system which is to serve something which cannot be completely defined. Eason argues that because current methods of task analysis concentrate upon the degree to which task closure can be assumed it is easy to overlook, or underestimate, the openness of a task.

Eason points out that the user is most likely to provide the adaptive component in a task performing system and is therefore a source of openness. The allocation of functions assumes closure, in the sense that certain aspects of a task are closed to the user (given over to the computer). Unfortunately, we do not know how important the various tasks are to the successful achievement of goals. It may be that although as person is not well adapted to performing a specific task, never the less success may depend, in some way, upon the person performing that task.

Allocationing functions to the man and computer based on their recognisable abilities means that we must have a good understanding of the design, or task, process if the closure resulting from the partitioning is not to be destructive. Clearly, the creative process is not well understood and consequently the value of the "allocation of functions" in this context is debatable.

Negroponte avoids the problem of closure resulting from the allocation of tasks by default. He appears to eschew the conventional belief that man is good at somethings and the computer is good at others, and proposes instead that the computer can be programmed to perform the tasks which appear on the "Man Good" side of the "allocation of function" table.

4.2 THE SYSTEM AS PARTNER/MENTOR.

Negroponte is well aware of the current limitations and difficulties of systems which are supposed to function as creative partners. He identifies three main problems with CAD systems; these being timing, the thwarting of creative leaps and paternalism. The problem of timing is concerned with the question of when you should provide hints so as to augment, rather than thwart, creative activity. The problem of thwarting creative leaps he argues, should be resolved by ensuring "the suspension of critical judgement in moments of collaborative efforts to find that which you are looking for... Premature intrusion of judgement aborts the ideas which could prove to be most valuable" (Negroponte, 1977). The last question, that of paternalism, is concerned with the way in which the computer reveals its intelligence to the user.

Negroponte believes that the solution to these problems lies in the idea of a personalised system. Work styles, he argues, are idiosyncratic and appear to become even more idiosyncratic the more creative the endeavour. Given that a persons work style develops over a lifetime and varies from person to person the problem, he argues, is how do we reflect these differences in the specific hardware and software of a CAD system. Essentially the answer is seen to be systems which learn to differentiate between users in a highly personalised way. Negroponte's vision is embedded in, and dependent

upon, the development of intelligent machines. Before long, he observes, CAD systems will emerge as, "idiosyncratic systems of the most ubiquitous sort, potentially the most widespread amplification of creativity seen by mankind..... Such romantic visions are important fuel for the daydreams of computer scientists and designers working on CAD" (Negroponte, 1977).

4.3 CONCLUSIONS

Gaines and Facey (1977) observe that a man adapting to a machine and a machine adapting to a man produces an inherently unstable system. Given the current state of the art both in terms of design process research and machine intelligence the kind of system proposed by Negroponte is bound to exhibit instability. For the time being, what he advocates must remain a "romantic vision". A goal to aim for perhaps, but one requiring a great deal of work.

On the other hand, the servant paradigm breaks down in respect of tasks which display openness. Eason (1979) points out that the most likely adaptive component in the system is the user who, in order to be adaptive must be able to interpret the task and select a method of execution; and who as a result of previous task effort is also likely to learn and change his approach to tasks. "As a result of these factors", Eason writes, "even in circumstances where the task is relatively unchanging, the demands upon a computer system may vary unpredictable because different users model the task differently and any single user changes his approach over time". This seems to imply that to cope with user openness the computer system should be adaptive; it should respond to the changing needs and demands of the user. If this is the case we are presented with a dilemma; we cannot do without adaptive systems and yet there is not enough information available for the system designer to construct adaptive systems which can function effectively with the openness of creative design.

And yet, it must be recognised that conventional artistic media are not adaptive. Oil paint, for example, does not respond to the user in an active way and yet who could have predicted, on its invention in the fifteenth century, that its use would result in such a remarkable variety of paintings.

Oil paint is a medium which, from the users standpoint has proved to be highly adaptable. Consequently, it is not a necessary condition of a computer system that in order to cope with open tasks it must be able to change in response to the user. An alternative approach, and the one adopted in this thesis, is to design a system in which the facilities provided can be adapted by the user so as to meet the requirements of his task. The characterisation of the computer system which results is one of a passive assistant who does only that which he is asked to do, without interfering or advising; and then only because by the nature of computer technology the user cannot perform some tasks directly.

CHAPTER 5

THE DESIGN PROCESS AND ASPECTS OF PLANNING

> "Every design problem begins with an effort to achieve fitness between two entities: the form in question and its context. The form is the solution to the problem; the context defines the problem". (Alexander, 1964)

The context which defines the problem here is that of the design process. The form representing the solution is, of course, a graphics system. Thus an analysis of the design process (context) should help to define the requirements to be met in the graphics system (the form).

5.1 DESIGN PROCESS RESEARCH

Design process research needs to be distinguished from research in methods of systematic design. Design process research is concerned with what the designer does whilst design method research is concerned with what he should, or might do. Cross (1977) observes that "unfortunately very little research on design process has been conducted". Cross reviews the work of a number of workers (Levin, Duckman, Maples and Gregory) studying architectural, industrial and engineering design. This work is not directly relevant because it is concerned primarily with aspects of three dimensional design, whereas this thesis is concerned with two dimensional design.

Mallen (1974) argues that man has developed a sophisticated interface between himself and other natural systems, this consisting of his tools and technology. Two types of interaction may take place ; the first between the man and his environment and the second between man and the interface. Corresponding to these two sorts of interaction there are two sorts of design behaviour. The first Mallen calls "feedback" design which occurs when a need is perceived and a design is executed to satisfy this need. The second sort of design activity is concerned with the operation of

the "tool-technology" interface itself. "In particular", he argues, "its aim is to create and extend this interface", so as to increase the likelihood of it being able to deal successfully with unforeseen hazards. He calls this sort of behaviour "feedforward" design. Plotting various design activities on the feedback/feedforward dimension he places engineering and product design towards the feedback end, and fine art and architecture towards the feedforward end.

Certainly the artist and creative designer is not concerned, primarily with providing solutions to actual needs. Fashion textiles, for example, is the result of a manipulation of the tool technology interface which results in forms which exist before any need for them is expressed ; it is only when they appear in the shops that people demand them. The artist is more concerned with inventing problems rather than responding to them.

Most design process research has been concerned with the "feedback" end of design and as a consequence is not useful for making inferences about "feedforward" design. An alternative approach to the problem, which is adopted here, is to make inferences from an examination of the work of artists, and their writings.

5.2 ASPECTS OF PLANNING

An important requirement of the man/computer interface is that it should be flexible enough to cope with the highly idiosyncratic working processes of artists. One factor which contributes to, and modifies, an artist's interaction with the medium is the degree of planning which characterises his working process.

5.2.1 PRE-PROGRAMMING

Over the last fifty years an approach to picture making has been developed which is characterised by the use of generative procedures. This movement has been particularly influential in Europe and England, where it is championed by the Systems group. Historically the work of the *systems* artists has its roots in Constructivism which developed in Russia during the first quarter of the 20th century.

The Constructivists renounced all associations with the depiction of natural phenomena and attempted to create an art of *pure* or *absolute* form, figure 12. Using, in general, a repertoire of geometrical forms the Constructivists constructed their works using industrial materials such as metal, glass, and plastic. Ideologically the Constructivists were in sympathy with the workers revolution in which many artists were actively involved. They aimed to democratise art ; to destroy its elitist, middle-class associations and make it accessible to everyman. This led to an industrialisation of art (not only in terms of the materials used but also in their construction) which in turn led to a depersonalisation of the picture making process and the separation of production and formulation. In this event the artist functioned rather like a designer by specifying a product which might then be produced by someone other than the artist.

Although for different reasons perhaps, many of the ideas proposed by the Constructivists have remained valid currency. The French painter Francois Morellet (1974) writes of his work, "From 1952 onwards I have been making useless, therefore, artistic objects with the constant purpose of reducing to a minimum my arbitrary decisions. I have supressed composition, removed all interest in execution and above all I have applied rigorously systems which are both simple and obvious".

Morellet's use of procedures seems to be prompted by a desire to remove decisions that might occur arbitrarily during the production of a picture. Different artists use systems, or plans, for different reasons but in all cases they are involved in the pre-planning or pre-programming of the picture making process.

This involvement in planning has tended to locate this type of artists interest in the process. Consequently an important part of his activity is the specification of procedures for generating art works. Jeffrey Steele, one of the founders of the Systems group in England writes that "A works structure is a coherent stage in an ordered sequence of operations each of which can be flexibly expressed within·determinate rules" (Systems Catalogue, 1972). In this sense the artist is a programmer, rather like the computer

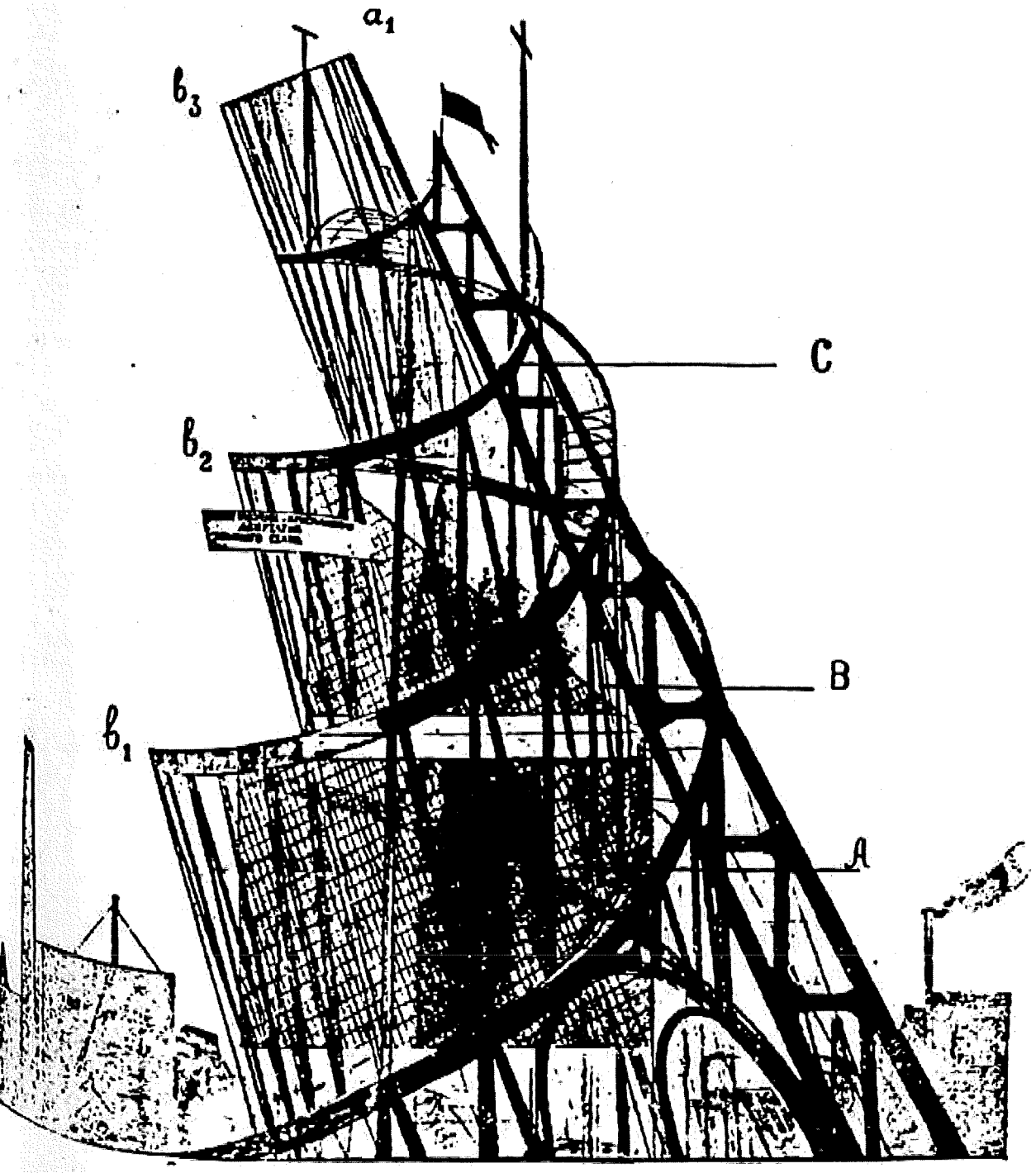


Figure 12. Tatlin, Tower or Monument to the Third International 1919-20

programmer except that he writes programmes for a human processor. It is not surprising, therefore, that many of the artists currently using the computer might be loosely described as *system* artists.

This type of artist represents the extreme case of *pre-programming* or *pre-planning*, the picture making process. Before the artist begins any productive acts he has pre-determined what those actions will be. At the other end of the scale there is the artist who, when first putting pen to paper, has absolutely no idea about what will follow except that by the nature of things some marks will occur.

5.2.2 MOVING FREELY

This approach to *starting* (in contrast to the systems artist who both prescribes the *start* and *finish* prior to commencement) is best illustrated by the working method of the Swiss artist, Paul Klee (1925), who writes "An active line on a walk, moving freely, without goal. A walk for a walks sake", figure 13. For Klee it was enough to start ; events being allowed to take care of themselves. Klee's method starts not with pre-planning but with a confrontation ; the confrontation of the artist with his medium. Whilst a strategy for beginning his approach in no way prescribes the events that will follow.

Thus we may predict at least two confrontations between the artist and his medium, which are characterised by the degree of planning involved. In the first case the confrontation marks the beginning of a procedure driven interaction, the second the start of an event driven interaction. Put in another way, the artist who pre-programmes his activity is able to do so because he knows what he is going to do, whilst the artist cast in the Klee mould does not know (often it is a deliberate strategy of suppressing such knowledge) what he is going to do and is therefore unable to pre-plan.

Clearly these represent the extreme poles of a single planning dimension with *all* at one end and *none* at the other. Klee develops his opening remarks in The Pedagogical Sketchbook by observing that "already at the very beginning of the productive art, shortly after the initial motion to create the first counter movement of

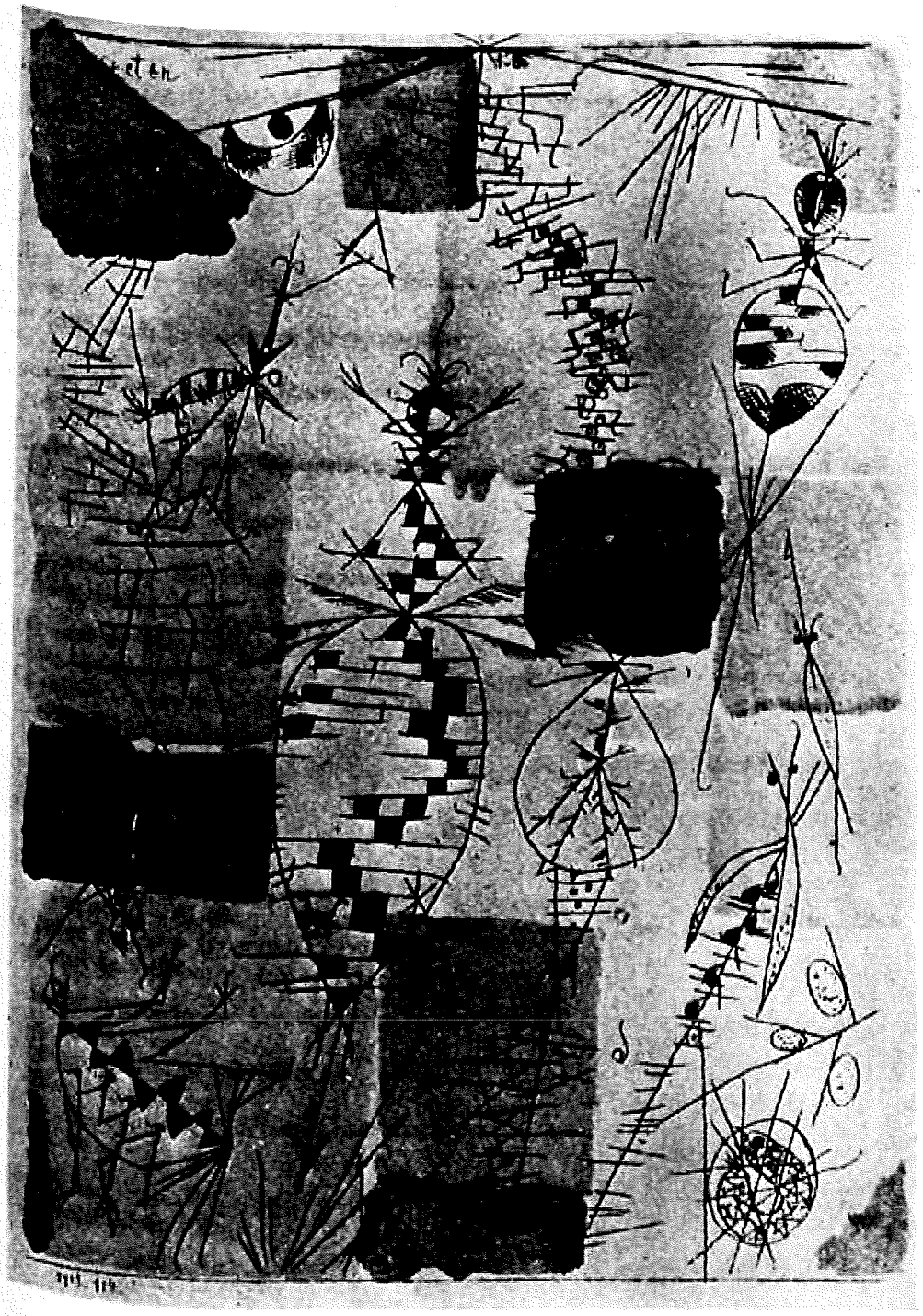


Figure 13

receptivity. This means that the creator controls whether what he has produced so far is good". Once the creator begins to make evaluative judgements about the work in progress control enters into the process, guiding the interaction and making planning possible.

5.2.3 PLOTTING THE PLANNING DIMENSION

If one is collecting flowers one way to do it is to follow a prescribed plan. For example this plan might way, "start at location A, proceed west fifty paces and if there is a flower nearby pick it and then turn left....etc.". Notice that this kind of planning is concerned with how to proceed. It guides the producer but it does not predict the goal. Another type of planning is that which is goal oriented. If one wants to collect a specific bouquet of flowers the problem becomes one of defining the process in terms of the goal. One might way that the first form of planning is *event* (action) orientated whilst the second is *goal* (object) orientated. Both type of planning are likely to occur as part of the picture making process. Alternatively, let's suppose that one has no procedures to follow and no goal to seek then all one can do is to start. For example, one might just drive out into the country, park the car, get out and start collecting flowers. This is analogous to the method that Klee describes. As Klee points out, having started the events which follow will soon be tempered, and controlled, by the processors response to the processed. However, the *random walk* should not be considered simply as a way of starting in the absence of creative ideas. Instead, it should be regarded as a creative idea in its own right. The event and goal driven process place constraints on the interaction between the creator and the object in progress. Returning to the problem of collecting flowers, it is as if they provided blinkers for the flower picker to wear. The blinkers help to keep the flower picker on the right path by reducing his view of the world, and the likelihood of him noticing more interesting but perhaps less well trodden paths, off to his right or left. Thus the *random walk* may be viewed as a means of discarding the blinkers of a preferred path thereby revealing a broader landscape. By making the decision to wander the flower picker confronts a world full of possible paths, some of which may prove to be uncharted and unexplored.

Figure 13

Typically the picture making process will be a combination of wandering and purposeful activity. Writing of his working process the painter Matisse has this to say. "Suppose I set out to paint an interior : I have before me a cupboard ; it gives me a sensation of red- and I put down a red which satisfies me ; immediately a relation is established between this red and the white of the canvas. If I put a green near the red, if I paint in a yellow floor, there must still be between this green, this yellow and the white of the canvas a relation that will be satisfactory to me" (Guizhard-Meili, 1967), figure 14. This quotation reveals a complex feedback process between the artist and the work in progress. Frequently changes in direction take place in response to the developing work. Sometimes these changes of direction will appear almost irrational with the creator taking one step at a time :-

> Why did you make that change?
> Because I did.
> But what did you see?
> I saw that it should be changed.
> Well if you change it, what was wrong with it before?
> Nothing, I tend to think one thing is as good as another.
> Then why change it?
> Well, I may change it again.
> Why?
> Well, I won't know until I do it. (Crichton, 1977)

At other times changes of direction may be accompanied by planning. Whatever else, the picture making process can be extremely complex in its directional meanderings.

Thus if we add to our axis of planning a time axis, the picture making process of different artists (or even the same artist) might generate, for example, graphs characterised by straight lines (at one extreme or the other) or meandering lines up and down the planning axis. While the exact nature of the planning/time graph is not important here, clearly the fact that the planning aspect (that is the number of steps that can be planned in advance, rather

like moves in a game of chess) can vary considerably during the creative process has important implications on the design of the interface between the man and the computer.

At one extreme we have the artist who knows what he wants to do and is able to describe his activity in a procedural way. Clearly, the computer is going to be more accessible to this kind of artist because his working processes may be expressed algorithmically. It has sometimes been suggested that only the artists who knows what he wants to do can make any use of the computer. Jasia Reichardt (1971) has written "with computer art, at least that which is produced by the computer, the artist must know what he wants to do". However, this is not an incontravertible law it is rather a consequence of limited resources. As we have seen, at the other extreme we have the artist who doesn't have an objective in mind when he begins to work and clearly his needs are much more difficult to satisfy because they are difficult to predict. If we want to extend the creative use of the computer we must provide facilities which bridge the gap between the "knows" and the "don't knows".

For the *knows*, since they are able to make plans a prescriptive language is indicated ; one which allows them to define procedures. As illustrated before, these procedures will be of two kinds ; event orientated (action) and goal orientated (object) procedures. The action procedures will control the execution of the picture making process whilst the object procedures will generate particular structures. For the *don't knows* an interactive language is essential since each step is conditioned by the response to the results of the previous steps.

For the majority of artists the productive process is a combination of *knowing* and *not knowing*. Thus at times he will be taking one step at a time and at other times leaps of varying distances, the length of a leap reflecting the degree of planning, or programming, undertaken. Consequently in order to bridge the planning dimension we can predict the need for an interactive language with a structure which permits the user to express various levels, and types, of advance planning.

As indicated previously, in the Matisse quotation, the picture making process can be modified as a result of feedback from the picture as it develops. In the next chapter the significance of the picture as it is perceived by the artist will be examined.

CHAPTER 6

THE IMPLICATIONS OF THE UNCERTAIN IMAGE

The previous chapter argued that "wandering", that is to say starting without any preconceived object, is not a negative process but a creative strategy. Essentially it is a device for maintaining a state of flux which inhibits the tendency to make critical judgement at too early a stage in the creative process. In the context discussed previously *wandering* referred to a procedure ; to how the artist or designer proceeds. However *wandering* may also be shown to have significance in relation to the perception of the object being produced. In other words the artist may employ techniques which keep the perceived image in a state of flux, also. Furthermore the perceptual ambiguity which results from indeterminant stimulii can be shown to be an important aspect of creative design.

6.1 THE UNCERTAIN IMAGE

Most of us have experienced, at some time, foggy or misty weather conditions. Without such phenomena the thriller film would be robbed of one of its most useful devices for creating suspense. The scene where the hero, or heroine, is shown in the middle of a "pea-souper" staring wide eyed into the swirling mist trying to decide whether the indeterminant shape in the distance is a man, woman or beast, is commonplace. It may be a cliche but it does recognise the fact that when the edges of reality are, at it were blurred, the power of the imagination manufactures its own world.

6.1.1 MANUFACTURING THE INDETERMINATE

In real life, of course, foggy weather may be merely unpleasant, or even hazardous. After all, a world which appears ambiguous can be frightening ; we like the world to appear stable and consistent For the artist, however, ambiguity and illusion may be the source of creative ideas and invention.

In writing of the technique called Frottage, which he invented, Max Ernst (1948) has this to say, "I made from the boards a series of

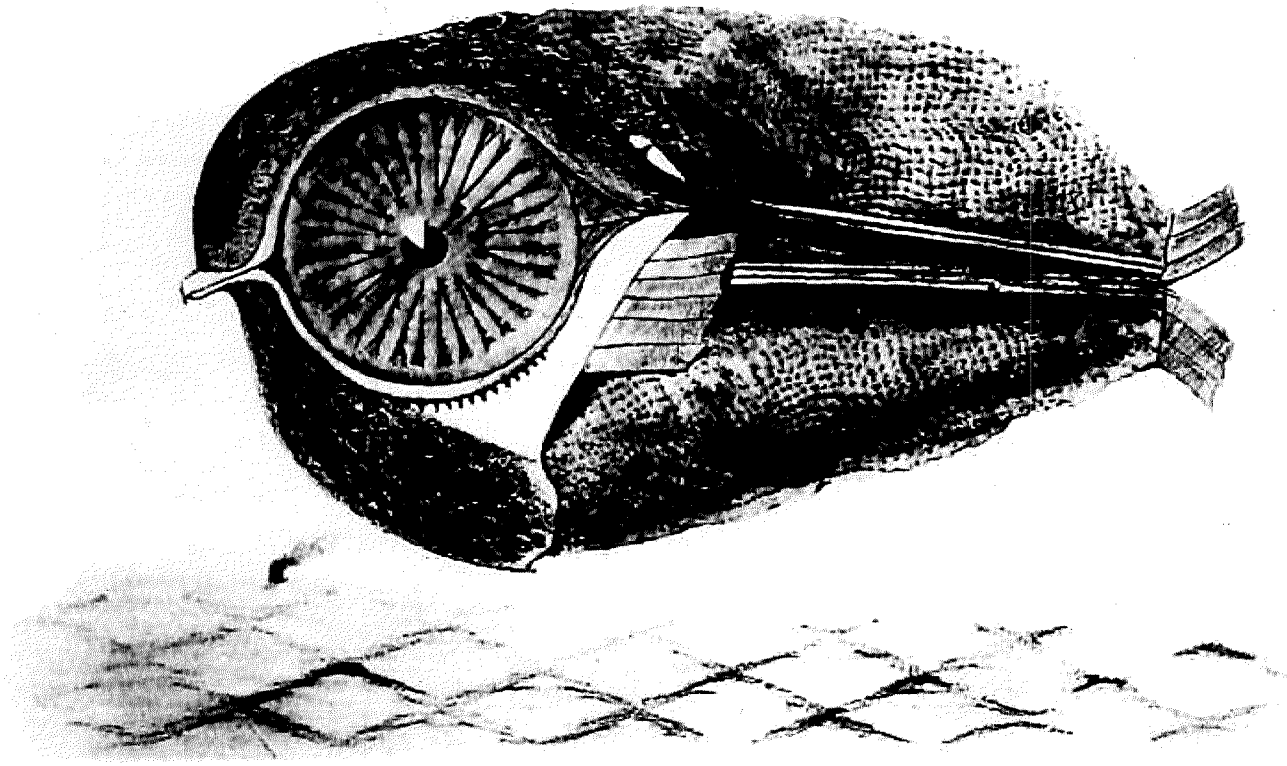


Figure 15

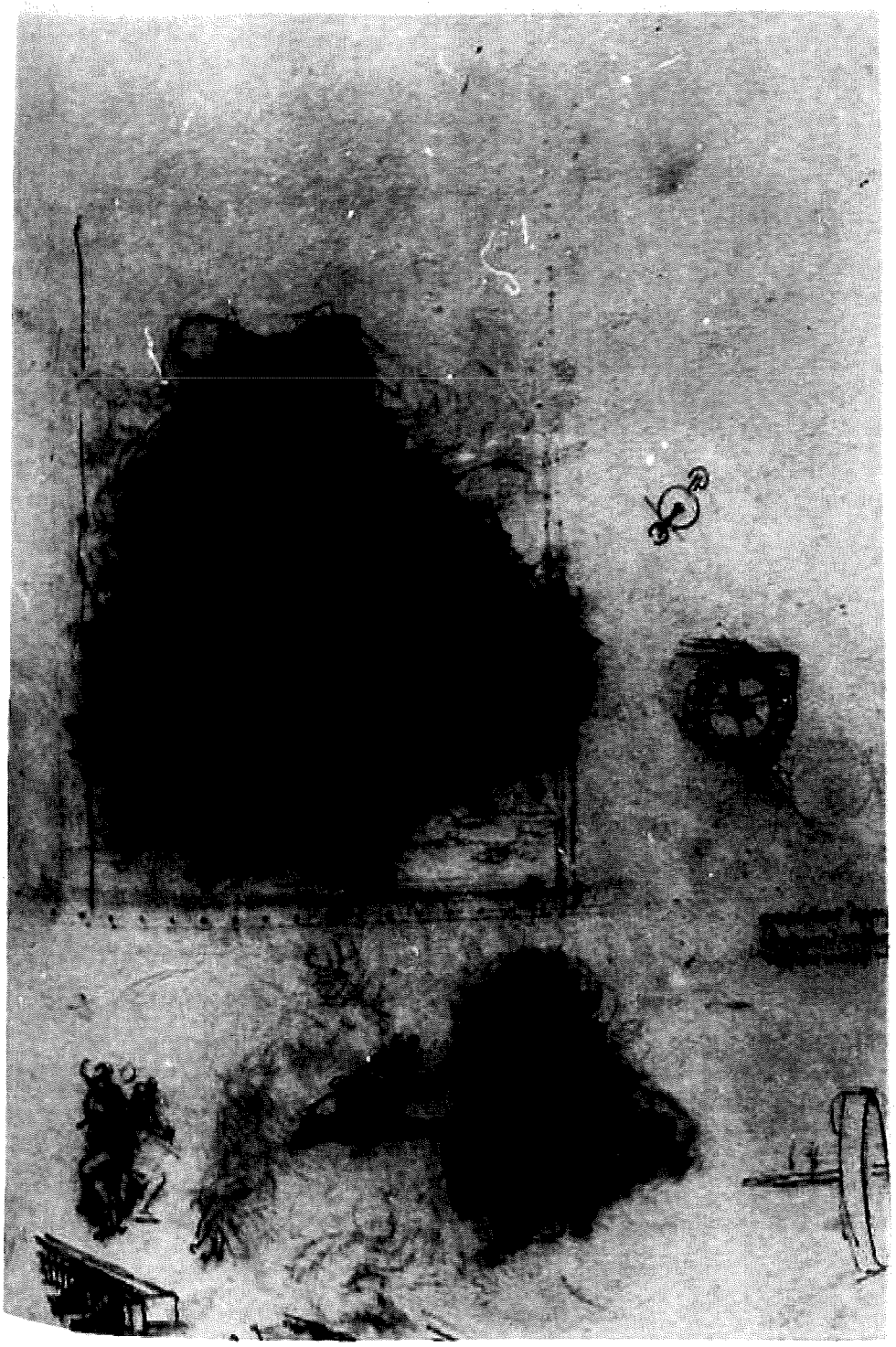


Figure 16

Figure 17

drawings by placing on them at random, sheets of paper which I undertook to rub with black lead. In gazing attentively at the drawing obtained, 'the dark passages and penumbra', I was surprised by the sudden intensification of my visionary capacities and by the hallucinatory succession of images superimposed, one upon the other with the pesistence and rapidity characteristic of amorous memories".

Here Ernst is describing the use of a procedure (one determining how to proceed rather than how to achieve an objective) which produces marks open to various interpretations. During his lifetime Ernst was to produce many techniques aimed at presenting the eye with undifferentiated marks which the imagination could clothe with substance, figure 15.

6.1.2 THE IMPORTANCE OF THE INDETERMINATE

In the above passage Ernst quotes Leonardo who was perhaps the first artist to recognise the value of productive techniques which keep the image in an indeterminate state during the early stages of picture making. Leonardo's sketches are characterised by 'pentimenti', a welter of lines which shatter the integrity of the perceived image, figure 16. Prior to Leonardo, drawing was used mainly as the testing ground for the picture composition or as raw material for the details of a picture. The drawings of Pisanello, for example, figure 17, are characterised by their precision ; by their hard edged quality. When the pen was put to paper, for Pisanello it was an act of commitment ; a fact rather than a possibility. Leonardo, on the contrary, believed that a drawing should not be articulated by means of precise outlines since this tends to prescribe to early the finished picture, thus making it difficult for the artist to modify his ideas.

Gombrich (1966) in discussing Leonardo's use of the sketch has observed that for him the indeterminate rules the sketch as a means to stimulate the mind to further inventions. He cites an example which clearly demonstrates how a picture perceived by Leonardo in an alternative way provided the stimulus for a new composition. Gombrich compares the Neptune sketch (figure 18), which was produced whilst Leonardo was in Florence engaged on the painting of the

"Battle of Anghairi", figure 19, to the latter arguing that the figure rising with upraised arm over a group of horses suggested the image of Neptune driving his sea horses. As Gombrich puts it, "In searching for a new solution Leonardo projected new meaning into the forms he was in his old discarded sketches".

The following quotation (McCurdy, 1910) also reveals the value Leonardo attached to perceptions resulting from the study of patterns.

"I will not refrain from setting among these percepts (Devices for Painters) a new device for consideration which, although it may appear trivial and almost ludicrous, is nevertheless of great utility in arousing the mind to various inventions. And this is that if you look at any walls spotted with various stains of a mixture of different kinds of stones, if you are about to invent some scene you will be able to see in it a resemblance to various different landscapes adorned with mountains, rivers, rocks, trees, plains, wide valleys and various groups of hills. You will also be able to see diverse combats and figures in quick movement, and strange expressions of faces, and outlandish costumes, and an infinite number of things which you can reduce into separate and well conceived forms. With such walls and blends of different stones it comes about as it does with the sounds of bells, in whose clanging you may discover every name and word that you can imagine".

Likewise Ernst makes use of techniques employing chance to generate indeterminate forms open to alternative perceptual interpretations. In a sense, these devices present problems, in the form of surfaces textured by marks, for the visual sense to solve. A degree of uncertainty in the visual stimulus is important because it results in many, equally plausible perceptual solutions and thus promotes an imaginative exploration of the image that may lead to novel interpretations. This stresses the creative significance of the image (the marks on a surface) as a perceived object.

6.2 THE PICTURE SURFACE AS PERCEIVED OBJECT

A concern for the nature of the picture surface as perceived, whilst

Figure 18

Figure 19

evident in earlier art (formal or structural aspects in picture making have always been of interest to the artist) is a particular feature of twentieth century art. As early as 1890 the painter Maurice Denis wrote, "Remember that before being a war-horse, a nude or some story or other, a picture is essentially a flat surface covered with coloured pigments arranged in a certain order" (Gilson, 1957). This statement is often regarded as the herald of a movement which was to transform art.

Cezanne, at the turn of the C19 developed an art form which stressed the integrity of the picture surface. Whilst still depicting objects in 3D Cezanne managed to preserve the picture surface. Paradoxically, when looking at his paintings one is aware of both the spatial dimensions of the objects depicted and the reality of the marks on a two dimensional surface, figure 20. During the next fifty years the subject (i.e. the story, the objects depicted) in painting, was, more or less, discarded in favour of a more abstract formal art.

The Suprematists argued for an art of pure form in which painting was completely free of subject matter. More recently artists such as Stella and Morris have sought to present the art work as an object in its own right ; subject presumably to the same laws of perception as any other object confronted in the real world, figure 21. Contemporary developments place the centre of interest, for the majority of artists, squarely on the formal and abstract properties of the picture. The process of abstraction in modern art has led to a greater recognition of the picture as a perceived object on its own right. Consequently, alternative perceptual interpretations of the picture are more likely to override the conceptual scheme represented by the marks

6.3 ILLUSION AND CREATIVITY

The psychologist, Richard Gregory's particular view of visual perception has led him to study in some detail the nature of illusions. Perceptions, he would argue, can be viewed as hypotheses and the process of perception one of hypothesis-building based on inferences from sensed data and stored memory. With this in mind,

Figure 20

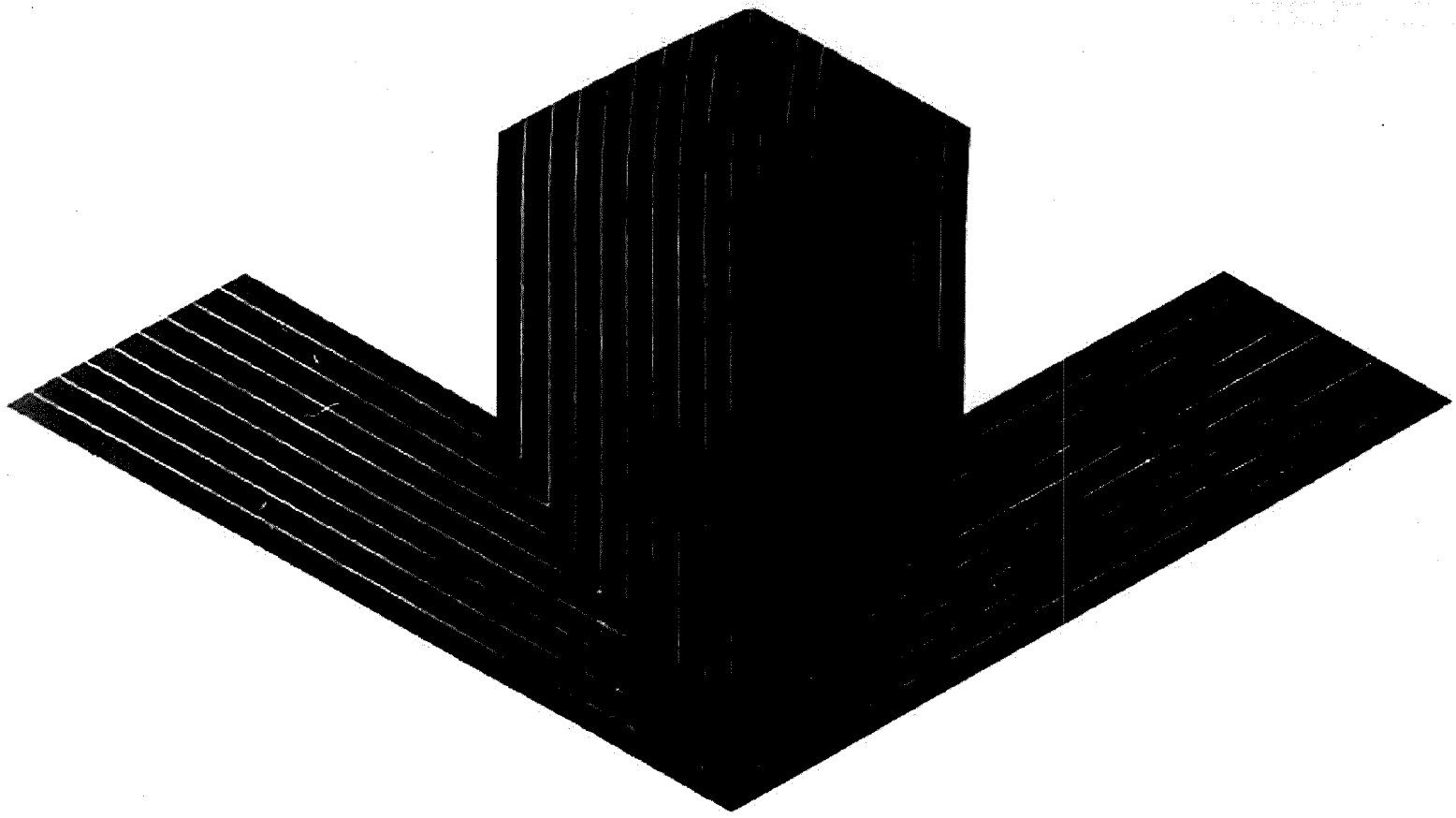


Figure 21

illusions may be regarded as errors generated by misplaced 'hypothesis-building' and 'hypothesis-selecting' strategies. "It may be", Gregory (1973) writes, "that susceptibility to illusion is necessary to being creative, for if we were controlled directly by sensed events....we would surely be tyrannized by the here and now - imprisoned by what is. Artists, with their skill, somehow play upon our potentials for illusion, allowing us to see and invent new possibilities. Perhaps they are extending in us what we routinely do to see objects in the real world". So the artist makes use of illusion to invent novel perceptions. But the artist is subject to the same problems of 'set' thinking as anyone else and thus it is not easy for him to view the world in a different way. To do this the artist often creates techniques which generate the unexpected, producing undifferentiated patterns open to a variety of interpretations ; one might almost call them illusion generators.

Gregory (1973) observes that, "Any perception clinging totally to what was, must fail just so far as the present differs from the past it represented. This can only be avoided by continually inventing a fictional future - which must be partly novel and may be true - if illusions of perceptual inertia are to be avoided. In these perceptual processes we may see the sources of discovery and invention. Here also, we see the artist playing upon and evoking our powers to invent other worlds". This is in agreement with Mallen's view of feedforward design which functions to extend the tool/technology interface. We can now see that methods (like those used by Leonardo and Ernst) provide the artist with the means of avoiding 'perceptual inertia' by creating a stimulus that promotes, in a positive way, the imagination via the perception of illusions.

6.4 THE PICTURE SURFACE AND THE ARTIST

The source of the illusion and the centre of the artist's attention is the object in progress ; the pattern on a surface. It is this concern with the picture surface as a perceived object that, in general, separates the artist from the engineering, architectural and product designer.

Drawings used by the architect, for example, represent the underlying structure of the design and are symbolisations of conceptual structures. When the architect generates and manipulates his drawings he is, in fact, operating on the structures they represent. Thus if he modifies four lines representing a room he is engaged in room arrangement not line arrangement. In this context the drawings may be viewed as an external memory which aids the designer in manipulating the design. In this case the focus of the designers attention is a conceptual framework that is represented in various information structures, such as drawings.

Consequently, the architect responds to the meaning of a drawing rather than its physical or perceptual properties. This does not imply that the architect is insensitive to the "aesthetic" qualities of his drawings but it does indicate that they will, in the end, be overriden by the demands of solving the architectural design problem.

In contrast, it has been argued that for the artist the vagaries of the perception of a drawing may result in the meaning attached to it being discarded in favour of a new interpretation. Indeed it has been suggested that an openness to the drawing as a source of possible interpretations can be a significant factor in the artistic process. The following chapter argues that the perceptual uncertainty of pictures has important implications for the design of computer facilities for picture identification and manipulation.

CHAPTER 7

PICTURE STRUCTURING AND MANIPULATION

One area in which the computer can augment or at least assist the artist is that of picture manipulation. For some time a picture of the designer effortlessly moving design elements about on the screen has been drawn in the literature, indicating the advantages of the computer graphics system. Clearly there is potential here ; if it is possible for the artist to manipulate and modify the picture in progress more effectively than existing media then computing begins to look a more attractive proposition.

The previous two chapters illustrated how tentative the artist can be, both in terms of his working process and in the interpretation of the picture as it develops. Negroponte (1977) points out that there is general agreement amongst writers on creativity as to the "Jekyll-and-Hyde" nature of judgement and imagination, which demands that the critical mind be suspended lest it hinder the production of ideas. A factor which plays a significant part in maintaining ideas in a state of flux is the medium used by the artist to express his ideas. For the artist must be able to change his mind about the work in progress and if the medium is such that it will not permit the modification of the picture then the statements he makes using it can hardly be regarded as tentative. Even though ideas may be tentative from the artist's standpoint an inflexible medium can be said to fix them for him.

Consider for example oil painting ; it is in fact a very good medium for the expression of tentative ideas. Being slow drying it is quite easy to erase or modify a picture by taking a rag or a palette knife to it. However, in general, such procedures for modification are crude and may affect other, satisfactory, parts of the picture. Thus to take a turpentine rag to an oil painting requires a certain degree of courage since it is quite easy to spoil the picture, the fear of which can inhibit tentative decision making. One tends to want to be sure that the modification will lead to improvement before making it. Also such procedures are negative since after an erasure a large degree of re-generation may be required. If an artist

decides to move part of a picture then he must erase and redraw. How much easier it would be if all he has to do is to pick up the appropriate part and reposition it.

7.1 STRUCTURING THE PICTURE

Since any manipulations of a computer generated picture that an artist might want to perform must take place indirectly (via a man-machine interface) both the man and the machine must share a common model, or structural representation of the display picture. Thus if the artist asks the computer to, "move this square to this position", the computer must know (at some level of understanding) what a square is, and must be able to locate a particular square at a particular location in order to move it to a new position.

In conventional computer graphics systems, picture manipulation is made possible by facilities that the user can employ to communicate his structural interpretation of a picture to the computer, figure 20. Essentially what happens is that he builds up a description of the picture inside the computer using the structural and pictorial concepts that it has been programmed, or designed, to understand. In general, the displayed picture is then generated from this internal representation of the picture. Typically, additional facilities are provided (i.e. move a shape) with which the user can modify the computer's internal representation of the picture and consequently alter the displayed picture. There are two important points to note here : although the user may think that he is working on the displayed picture he is in fact operating on the computer's internal representation of it (we shall return to this point later) ; also description precedes visualisation which precedes manipulation. For this reason the process of describing a picture to a computer system in this way is defined as "pre-generative structuring".

A number of examples have been cited (Ernst, Leonardo) of a structure being perceived in marks *after* they have been generated or found. We can imagine that having perceived a new structure the artist may wish to communicate it to the computer. Whilst *pre-generative structuring* takes place *before* the picture is generated (at least in the mind of the artist) the structuring activity mentioned above

occurs *after* generation and is applied to a picture which has no previously defined structure. Consequently, it is defined as "post-generative structuring". Ernst (1948) described how he perceived a "succession of contradictory images superimposed, one upon the other". Consider for example figure 23a which could be perceived as 23b, 23c, 23d or 23e.

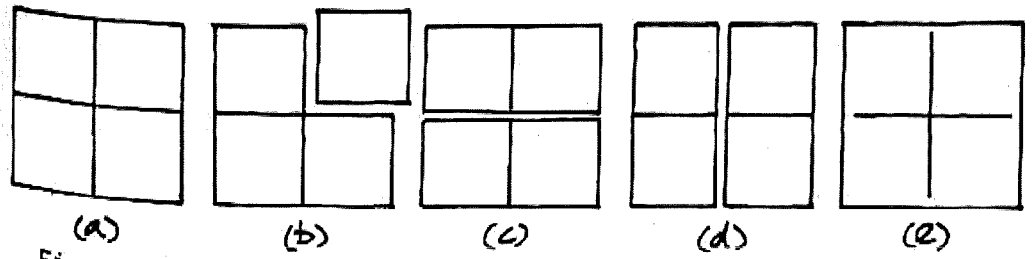


Figure 23.

There are, of course, many other possible interpretations of the figure. The artist, in perceiving a new figure in one which is already represented in the computers memory may wish to restructure it. Restructuring is distinct from *post-generative structuring* because although both are applied to a picture which has been generated, the latter involves the redefinition of a defined structure whilst the former relates to a perceived shape which has no previously defined structure. For this reason the latter is defined as *post-generative restructuring*.

7.2 EXISTING STRUCTURING FACILITIES

During the last fifteen years or so, the principal devices for interactive computer graphics were storage and refresh screens. In the following pages the structuring facilities provided by these devices will be examined and their success in terms of the structuring activities described in section 7.1. determined.

7.2.1 THE STORAGE SCREEN

The storage screen is the simpler of the two devices from the point of view of its processing power. Also it is relatively inexpensive by comparison with refresh displays. In brief, it operates by intensifying a grid which in turn bombards the screen with electrons which cause its phospher coating to glow (Newman, Sproull, 1973).

Once intensified the image remains bright until it is deliberately erased or until leakage from the flood of electrons through the collector grid intensifies the entire screen. The storage screen has no effective knowledge of what is displayed on it. If the screen image is to be manipulated then the graphics system must have a representation of it. If no internal data structure exists then the picture cannot be manipulated. Few storage screens permit selective erasure and consequently if any part of the displayed picture is modified the entire picture must first be erased and then redrawn by tracing through the updated picture representation. This lack of selective erasure has proved to be a major drawback of storage screens in interactive graphic applications.

7.2.2 THE REFRESH SCREEN

The refresh screen on the other hand produces an image which quickly fades. Consequently, in order to produce a stable image the screen must be refreshed (redrawn). This refreshing process must be repeated around 30 times a second in order to avoid flicker where the image appears to blink. The early refresh displays were called "point plotting displays" because lines were displayed as collections of points, each point being distinctly visible on the screen when closely scrutinised. This type of display was driven by the computer to which it was interfaced but this proved to be impractical because the limited speed of computers restricted the number of points that could be redrawn without flicker, and also because the computer was dedicated to the menial task of displaying the picture when it might have been better employed, for example, interacting with the user. Fortunately hardware developments solved these problems. In the first place the refresh display was given a certain degree of autonomy from the host machine by building into the display a processor (often referred to as a display processor) which took over the function of displaying the picture on the screen by accessing a picture memory called a display file. This display file is effectively a program for execution by the display processor which exhibits many of the features of a conventional computer program- it may for example include GOTO and SUBROUTINE instructions (Newman, Sproull, 1973).

The problem of image complexity was solved to some extent by providing hardware for generating graphic primitives such as a line or an arc. A vector might be represented in a display file as an instruction containing a code meaning LINE and the endpoints of the line. The information contained in this instruction is processed by the display processor which passes the endpoints to the vector generator that produces the line. Thus the display file whilst having instructions other than those for generating graphical data also contains an implicit representation of the screen as, in general, a set of lines.

An important feature of the display file is that it can be structured using the subroutine and transfer of control functions mentioned above. For example, sets of display file instructions can be isolated from each other using GOTO instructions. When the display processors starts the cycle of refreshing the screen it goes to the top of the display file and examines the first instruction. This might tell it to GOTO a location in the display file and continue processing from there. The next ten instructions might be drawing instructions which cause a circle to be displayed. At the bottom of these instructions there might be another GOTO which sends the display processor to the next set of drawing instructions and so on, figure 24a. The display file, broken up in this way, is called a segmented

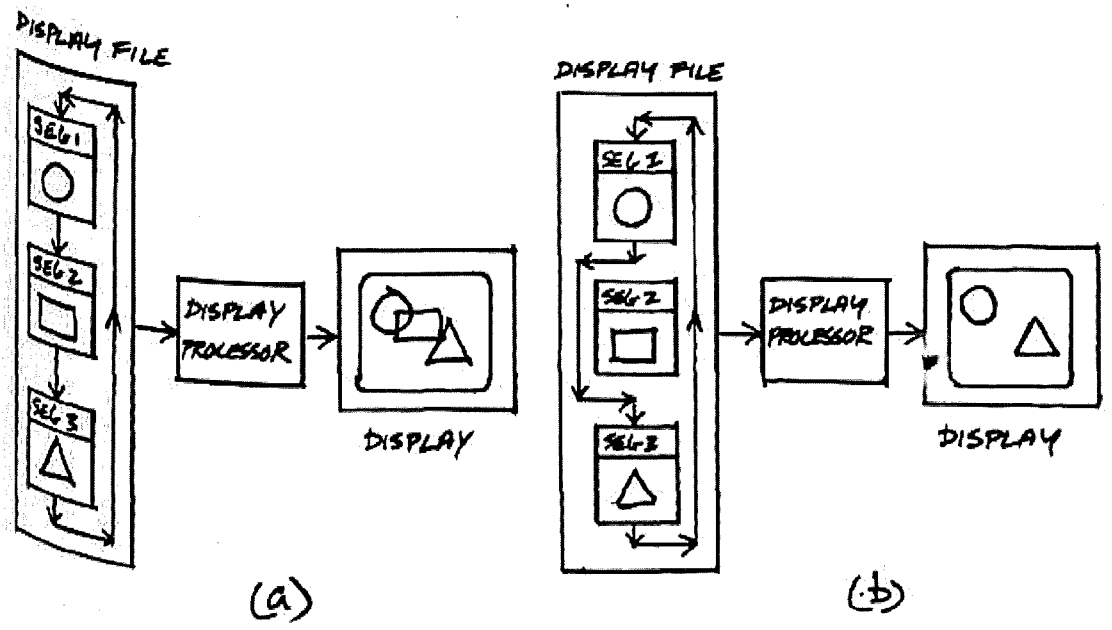


Figure 24 Segmented display file

display file. Although the display file is, implicitly, a representation of the screen it is not usually directly accessible to the user. In most computer graphics systems which use refresh screens, the picture generated by the user is first built up in a data structure. The graphics system then scans this data structure to produce display instructions which it puts into the display file. Consequently as the computer graphics system builds up display file segments (as prescribed by the user) it can keep a record of where they start, and if necessary where they end. Given this knowledge the segments in the display file can be manipulated by the graphics system. For example, selective erasure can be achieved by changing GOTO statements appropriately, figure 24b, causing a segment to be missed out of the refresh cycle, therebye causing it to fade and disappear from view. Selective erasure simply means that part of a picture can be selected for erasure without affecting the rest of the picture. In figure 24b the circle and triangle are unaffected. Later it will be demonstrated that whilst the ability to structure the display file makes it possible to manipulate parts of a picture it also results in an abstraction of the display picture which in some contexts is unsatisfactory.

7.2.3 PRE-GENERATIVE STRUCTURING

Pre-generative structuring is the structuring facility most easily provided in computer graphics systems. In storage display systems it is provided by data structure facilities and language operators which allow the user to build pictorial structures. Likewise, similar facilities are employed in refresh display systems.

7.2.4 POST-GENERATIVE RESTRUCTURING

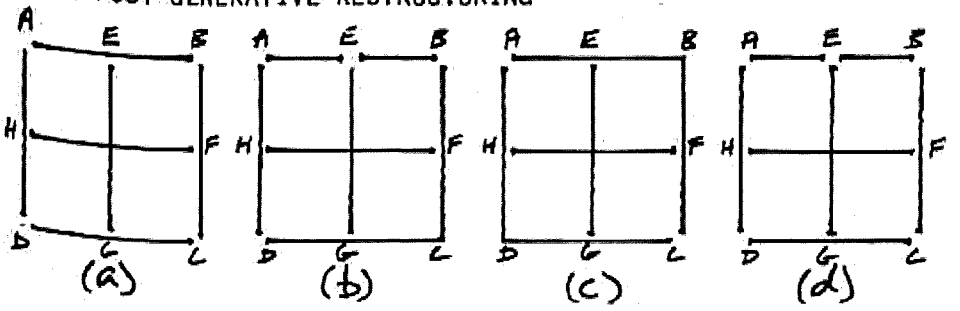


Figure 25

In general, computer graphics systems do not permit dynamic restructuring. Typically restructuring must be achieved by redefinition.

Rather than modifying only those parts of the structure that require it the old structure is replaced by a new one. There is in fact, no reason why existing structures such as figure 25.a. could not be restructured as figure 25.c.. Notice that although figure 25.c. has less components than figure 25.a. the number of vectors contained in the components remains the same and furthermore they are the same vectors. Whilst transformations such as that from figure 25.a. to figure 25.c., where the vectors remain unchanged, might be made possible by improving graphical data structures and the language operators for manipulating them, transformations such as figure 25.b. and figure 25.d. present a more difficult problem.

The reason for this is connected with the fact that in most graphics systems the basic graphical primitive is a line (vector). In the case of the refresh display this is the result of hardware developments aimed at solving specific display problems whilst with storage displays the use of the vector as the basic graphical primitive provides a way of saving computer memory. Whatever the reasons, the result is that graphic system designers have tended to think of pictures in terms of lines and so have provided facilities for organising and manipulating lines.

Given a picture such as figure 25.a. this would probably be represented by the endpoints of each vector. The restructuring resulting in both figure 25.b. and figure 25.d. cause the generation of two new vectors AE and EB which result from the division of the vector AB. If AB was represented by its endpoints the vectors AE and EB would not exist in this representation. Thus it is not just a matter of reorganising the graphical information in the data structure ; additional information must be generated which is not represented explicitly in the data structure. In practise acquiring this additional information proves to be difficult (if not impossible) and consequently facilities for restructuring are not provided on vector based computer graphics systems.

7.2.5 POST-GENERATIVE STRUCTURING

Likewise *post-generative structuring* is not possible when using a storage screen as a display. A picture is remembered, in a sense, by the storage screen but a computer graphics system cannot gain access to it. Therefore, if the computer system has no representation of the picture and no way of examining the contents of screen, then there is nothing to which it can directly relate any structured description of a perceived shape the user might communicate to it. Limited facilities for *post-generative structuring* are possible when using refresh displays by processing the display file. However, the same problems which make *post-generative restructuring* impractical apply in this case also. In short then, facilities which allow the user to restructure or apply a structure to a displayed image are not generally available when using conventional displays.

7.3 THE LEICESTER POLYTECHNIC RASTER SCAN DISPLAY SYSTEM

The conventional refresh screen is sometimes called a random display because a picture is displayed in an arbitrary way dependent upon the order in which the lines appear in the display file. Consequently the beam which is used to project the image on the screen can appear to traverse it in a random way (up to draw a line here, down to draw the next, left to draw the next and so on). Raster scan refers to the way that a television picture is generated by a beam which scans the entire screen line by line in an order which never deviates ; it always starts at the top line and works from left to right, returns to the left side of the screen and displays the next line and the next until the bottom line is reached, whereupon it returns to the top and repeats the cycle.

Raster scan display is the term used to describe a television display device connected to a computer. The raster scan display offers new possibilities in terms of the kind of pictures that can be displayed and manipulated. Random displays are primarily vector based and as a result are not readily adapted to handling tone or colour regions of any textural complexity, unlike the raster scan display which has the potential for generating and manipulating colour, tone and highly textured pictures because this is all within the scope of the

standard television.

Like conventional refresh and storage screens the picture to be displayed on the screen is stored inside the computer ready to be processed by the raster scan display. A special processor inside the raster display produces the screen image by processing the information contained in this display, or picture, memory. The display memory of a raster scan display is equivalent in function to the display file of a refresh display. Like a refresh display the picture memory must be processed around 50 times a second to prevent the picture flickering. Currently a number of techniques (Newman, 1978) are being employed to represent a picture in the display memory. The method adopted for the Leicester Polytechnic raster scan display is called a *bitmap*. In a *bitmap* every point position on the screen (the screen appears as a matrix of dots) is represented by bits of memory. Consequently it is a map of the screen matrix in which the physical state of each screen point (e.g. brightness) is recorded by bits of computer memory (sic *bitmap*). In the simplest case the number of bits per point is one where the number of point states that can be represented is two ; for sixteen levels of grey, four bits per point would be required. At present, the Leicester Polytechnic raster scan graphics display *bitmap* has one bit per point and two states each bit can take are used to represent black and white. The display is connected to a PDP8 mini-computer which can read from and write to the *bitmap*. Many raster scan displays do not permit information to be read from the picture memory by the host computer. Later, the usefulness of being able to *read* from the *bitmap* will be demonstrated.

In a vector based system (whether it be raster scan, refresh or storage) the graphical information is an abstraction of the screen. For example, the display file of a refresh display only contains the "drawn" lines ; those parts of the screen which are not "brightened" by its beam are not represented in the display file. Even the "drawn" lines are only represented symbolically as endpoints co-ordinates. In general, when using a graphics system the user really interacts with the computers internal representation of the picture, although he may think, subjectively, that he is operating

on the actual screen image. However, in cases where the user is working on what he *sees* in the displayed image a mismatch may arise between what is perceived and the computer's internal representation of the picture. As demonstrated, the information contained in vector based representations of pictures does not lend itself to *post-generative structuring* and *restructuring* and therefore the user and the computer are often unable to resolve mismatches. Earlier in this chapter it was argued that the medium used by the artist is an important factor in determining the ease with which tentative ideas can be explored. The inflexibility of vector based computer graphics systems with respect to picture structuring makes it extremely difficult for the artist to adopt a tentative attitude to the interpretation of what he sees in a picture.

The alternative methods for representing the picture for the raster scan display avoid the expensive memory requirements of the *bitmap* by what are basically data compression techniques. Consequently they have the same disadvantages as the vector orientated refresh and storage displays. However, since each memory location in the *bitmap* has a one-to-one correspondence with a point on the screen it provides a representation of the picture which is much closer to what the user sees. For example, if we regard "drawn" shapes as black and the "ground" as white, unlike random displays, the "ground" parts of the picture are represented explicitly in the *bitmap*. Also by using a *bitmap* it is possible to treat the point as the basic graphical primitive. Unlike the display file of a refresh display, the *bitmap* is not highly structured. It can be viewed simply as a matrix of points some of which are black and some white. Whereas in a display file graphical information is grouped together into segments, points in the *bitmap* are not grouped in this way. In a refresh system the grouping together of graphical information takes place as the display file is being added to. Once grouped together in a certain way it is difficult to restructure the display file. Earlier it was argued that this is partly due to the fact that conventional graphic displays do not store sufficient information in their internal representation of a picture for it to be easily re-interpreted. However, a *bitmap*, since it contains a lot of potentially useful information about the displayed picture and is not ordered in any way which might make extracting such information

especially difficult, offers considerable scope for the use of procedures which, by processing it directly, can be used to extract shapes perceived by the user.

7.4 SOME BASIC TECHNIQUES AND THEIR USE IN POST-GENERATIVE STRUCTURING AND RESTRUCTURING

In this section a number of basic techniques that process the *bitmap* directly are described. They can be employed to erase move and copy a class of shapes without the need for them to be represented previously in a data structure by the user.

7.4.1 FILLING-IN

Given a close shape such as figure 26.a. it is possible to fill the shape in therebye arriving at figure 26.b..

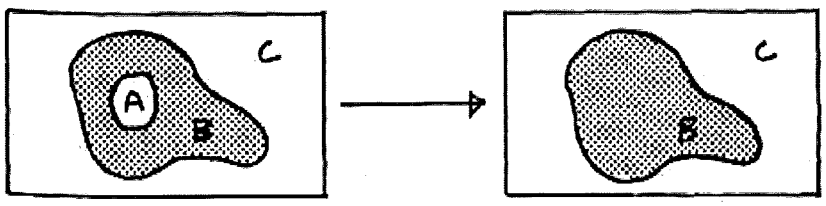


Figure 26 (a) (b)

This can be done by processing the *bitmap* directly and without the need for the shape to be defined in any data-structure. The process can be illustrated in the following way. A farmer wishes to clear an area in a field of grass. He does this by first cutting the grass to form a boundary (i.e. the boundary forming the inner white shape in figure 26.a.) around the area to be cleared. He then moves into the area and selects a blade of grass which he sets alight. This blade sets fire to its nearest neighbours, which in turn set fire to their neighbours and so on. In time the fire engulfs the area moving outwards to the cut boundary whereupon the absence of neighbouring grass causes the fire to burn out.

An algorithm based on this idea (Appendix 1) has been developed which will fill any region in the Leicester Polytechnic raster scan display system which is completely surrounded by its complementary

tone. The white circular shape A in figure 26.a. is completely bounded by points of complementary tone (black) and could therefore be transformed into figure 26.b. by the *filling-in* routine. The feature of this procedure is that it achieves the *filling-in* by processing (which involves reading bit values) the *bitmap* which it modifies as it proceeds. Furthermore it is a feature of the *bitmap* that terminates its activity (i.e. the complementary points which bound the shape being filled). There are, of course, hazards in using this kind of procedure just as there are for the farmer, in the analogy, who cuts a firebreak to contain and control the effect of his fire. If the farmer accidentally leaves a path of grass which connects the grass within the firebreak to that beyond the firebreak then the fire will spread along this channel and set the whole field ablaze. Likewise, if the shape to be filled is not completely isolated by complementary points the effect of filling-in can be disastrous.

Earlier it was pointed out that in conventional graphics systems the picture must first be given a structure before any manipulations can be performed on selected parts of it. Clearly, if the user must first describe to the computer the structure of a picture in order to operate on it then he must know beforehand what he intends to generate in order to assign it that structure. Furthermore if having communicated the structure of a picture to the computer, re-structuring proves to be out of the question then the user must have a clear idea of how he intends to manipulate the picture which must be considered when defining its structure. Obviously, with these constraints the user cannot afford to be tentative in his approach. Often systems are designed so that the explicit requirement of defining a pictures structure as one goes along is concealed from the user. For example, some architectural systems allow the designer to construct the plan of a building by selecting from a list of building elements such as walls, windows and internal parts. As the architect selects and locates elements the computer graphics system enters them into a data structure. Thus the architect is able to manipulate structural elements, such as windows, if the need arises. In this case the user does not himself specify the structure of a picture ; the computer system does it for him. In order to do this many assumptions must be made about what kind of

structures the user will define and how they will be manipulated. If these assumptions are wrong or incomplete the architect may suddenly find that he is not permitted to change his mind or perform a specified operation. Consequently, although the architect may regard his decisions as tentative he will discover, to his chagrin, that the computer regards them otherwise.

The *filling-in* process illustrates that it is possible, using a *bitmap* raster scan display, for the user to manipulate a picture without having to first describe its structure to the computer ; thus manipulation can precede description. Here then is an instance of a facility which allows the user to keep an open mind about the picture he is working on ; one which allows him to manipulate a shape as it is recognised.

7.4.2 MOVING AND COPYING

In a sense the *filling-in* procedure can also be said to recognise a shape belonging to a certain restricted class of shapes. Since all the points comprising a shape being filled are examined by the *filling-in* procedure it is a simple matter to modify the routine to record them. The modified *filling-in* routine now provides a way of *extracting* a shape from the *bitmap* and thus becomes a facility for *post-generative structuring* ; a way of describing to the computer graphics system a shape perceived by the user in the picture. Once an internal representation of an identified shape has been generated it is possible to move or copy it to a new location. The process of moving a shape is illustrated in figure 27. In the first place, figure 27.a., a point in the shape is identified by the user. This gives the *filling-in* routine its starting point and it then extracts the points comprising the shape, figure 27.b., at the same times effectively erasing the shape. Finally, the remembered shape extracted from the *bitmap* is redisplayed at a new user defined location, figure 27.c.. It follows from this that copying can be achieved by displaying the remembered shape at both its old and its new location.

7.4.3 SELECTIVE ERASURE

Filling-in can also be viewed as erasure. For example, if a black shape on a white ground was filled in then it would be effectively erased. Erasure can also be achieved by simply turning off points in the *bitmap* which are on (this is analogous to using an indian ink rubber). Used in conjunction these two methods provide a powerful selective erasure mechanism. Lets say that for some reason the user wishes to erase the roof of a house in figure 28.a. which has not previously defined as having a structure. One way of doing this is to first isolate the portion to be erased, figure 28.b. by simply turning the appropriate points off. The roof can then be erased using the *filling-in* routine. The same process can be used for moving and copying shapes connected to other shapes. For

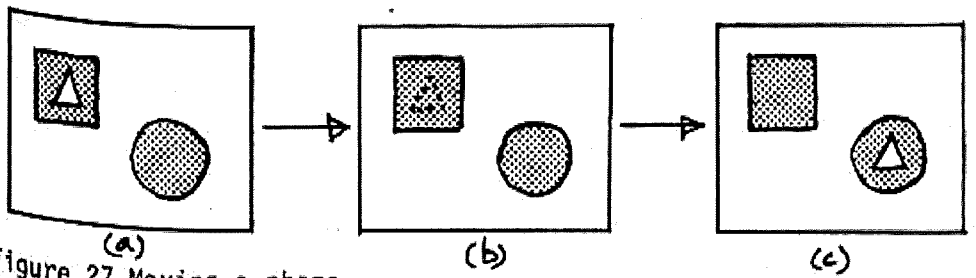


Figure 27 Moving a shape

example, instead of erasing the roof it could be moved to a new position. Photographs illustrating the use of these basic

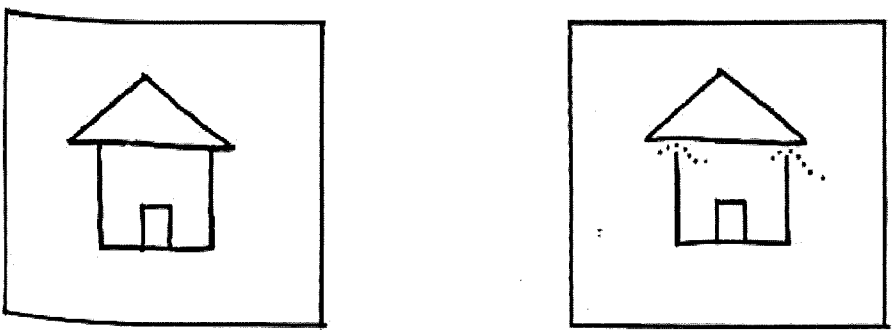


Figure 28 Erasing part of a shape

techniques can be found in Appendix 4.

7.4.4 THE *BITMAP* AS A SOURCE OF *PERCEPTIONS*

Clearly, by using procedures that examine and modify the contents of the *bitmap* it is possible to erase, move and copy shapes which have not been predefined by the user. These are all facilities which are only available to the user of a conventional refresh or storage display system if he has first described the structure of the shape to be manipulated to the computer. It is also clear that they provide facilities which in conventional systems require *post-generative structuring* or *restructuring*.

As the internal representation of a picture, in a conventional computer graphics system, is built up its parts (which might be a line, a square of a window) are identified as distinct elements. Some systems allow different levels of parts to be identified ; thus a square might be defined as four lines. The user can then refer to the square or to individual lines within the square. At the level of "square" the individual lines are related but at the level of "line" they are not. It is this notion of the picture as a collection of parts possessing different levels of complexity and bound together by a variety of relationships which makes it appropriate to describe it as having a structure. In this context *post-generative structuring* and *restructuring* imply processes of reorganisation resulting in non-trivial transformations of the structural description of a picture.

By contrast, the *bitmap*, which can be treated as functionally equivalent to the structural representation of a picture in a conventional system, has a structure in which all the parts (points) exist at the same level and are not grouped together by special relationships. It is interesting, therefore, to observe that when shapes are erased, moved or copied this does not involve structuring or restructuring the "bitmap". Thus after a "move", for example, the state of some bits have been modified but the structure of the *bitmap* remains unchanged. Consequently it no longer seems appropriate to continue regarding the manipulative operations described in the preceding sections as involving *post-generative structuring* or *restructuring*.

At this point a useful distinction can be drawn between the data structure of a conventional computer graphics system and the *bitmap* as representations of a displayed picture. The internal representation of a conventional graphics system can be regarded as the computer's interpretation, in a sense it's *perception*, of the displayed picture (or *visual stimulus*). Consequently when the users perception of the displayed picture differs from the computer's, it becomes necessary to change the computer's *perception* of the picture. It has already been argued that the structure of the computer's *perception* with its limited store of pictorial information makes certain changes practically unfeasible.

On the other hand, the "bitmap" can be regarded not as the computer's *perception* of the picture but as its *visual stimulus* ; as the source of its *perceptions*. To erase, move or copy a part of the picture can now be seen as a process of modifying this stimulus which may involve certain powers of recognition. For example, when a shape is moved the computer has to build a secondary representation of it and to do this it must therefore be able to *perceive* the shape in the *bitmap*. By taking this view of the "*bitmap*" the problem becomes not one of reorganising (structuring and restructuring) a complex structure, or *perception*, but one of recognising patterns (*perceptions*) in, and manipulating a data field, or *visual stimulus*.

In the context of interactive graphics the computer's powers of recognition and manipulation are not an end in themselves but the means by which the user achieves his purposes. Thus we should expect the computer's powers of recognition to be complementary to those of its human counterpart. In other words the user will want to be able to talk to the computer about what he *sees* in his own terms. In the chapter which follows facilities are described which allow the artist to identify and operate on pictorial properties of a perceived image.

CHAPTER 8

OPERATING ON PICTORIAL PROPERTIES OF THE IMAGE

Often what we think we see turns out to be an illusion. One common example of this is the illusion of water on the surface of a road on a hot summers day that is caused by the refraction of light rays as they pass through heated layers of air close to the level of the ground. The picture is perhaps the most extreme example of an object that causes us to see things which are not consistent with the physical characteristics of the stimulus. How ever convincing the portrait of a friend, or a landscape is, if we try to talk to the friend, or pick a leaf from a tree then we are bound to be disappointed. What we see does not stand up to close scrutiny. The kind of actions we might perform in response to the real objects depicted in a picture quickly reveal their incompleteness. A picture might deceive a bird, as did Zenxis' painting of grapes (so legend would have us believe) but a man is unlikely to be fooled for long. And yet the fact that we know a picture is a representation of the real world on a flat surface does not prevent us from seeing it as objects existing in a three dimensional space. Gregory (1970) has described the state of mind when viewing pictures as suspended belief. We allow ourselves to believe, for a time, in the reality of the pictured objects and magically the brain pieces together the marks on a two-dimensional surface and manufactures a three-dimensional space. Thus whilst Maurice Dennis was correct to assert that before anything else a picture is a collection of marks on a surface, typically what a viewer sees is much more. In fact it is extremely difficult to view marks on a two dimensional surface as just that. Almost involuntarily we begin to pick out objects and organise them in a world of their own just as Ernst did to such creative effect.

In the previous chapter it was argued that the problem of providing facilities that allow the user to manipulate a picture can be viewed as one of recognising shapes in the *bitmap* identified by the user. It was also pointed out that to be successful in this respect the computers powersof recognition should be complementary to those of its human counterpart. The *picture* as far as the user is concerned

is the screen whilst for the computer it is the *bitmap*. The problem then, becomes one of devising methods which allow objects to be extracted, or recognised, from the *bitmap* that are consistent with those identified by the user in the screen picture. However, as has been illustrated, whilst the picture is objectively a collection of coloured marks of different hue and intensity the user may subjectively experience the picture as a battle of horses and men. Thus the user may wish to manipulate aspects of a picture that are not objectively determinable physical properties. These can be called *pictorial properties* in contrast to the *physical properties* of a picture.

From the computers point of view it only has direct access to the physical properties of the *bitmap*. If it is to assist the user in operating on pictorial properties of the picture then these must be derivable from the physical characteristics of the *bitmap*. In the following pages some physical properties of the *bitmap* are identified and later it is demonstrated how information about the physical state of the *bitmap* can be utilised with information provided by the user in manipulating pictorial properties of the perceived picture.

8.1 THE REGION VIEW OF THE SCREEN MEMORY

In the previous chapter a number of facilities were described for processing the *bitmap*, such as filling-in. These facilities are possible because the *bitmap* has physical properties which can be inspected and altered. For example, erasure is an operator which works on the intensity value of points in the *bitmap* by setting any point indicated by the user to zero. Intensity is an observable property of a point which can be examined and processed.

The algorithm for *filling-in* not only makes use of the properties of individual points it also takes into consideration relationships between sets of points. More specifically, the filling-in algorithm makes use of the fact that any point can be said to be adjacent to other points in the *bitmap*. Any two points are said to be *adjacent* to one another if they are next neighbours. The next neighbour relationship is illustrated in figure 29. A pair of points of the same intensity value are said to be *connected* if

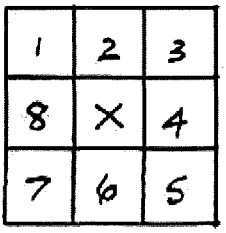


Figure 29 Next neighbour relationship

a path of adjacent points of the same intensity exists between them. It can be seen, by examining the *filling-in* algorithm, that given an initial point, adjacent points of the same tone as the original are iteratively processed until all connected paths eminating from adjacent points have been processed. In fact, what this algorithm processes shall be referred to as a region. A *region* is defined as a set of points of the same intensity such that all points in the set are connected by points in the set. The *bitmap* viewed in terms of regions can be described as a *region-map*, figure 30. Within a region two types of points can be identified : boundary and non-boundary points. A *non-boundary* point in a given region has adjacent points which are all in the region under consideration. A *boundary* point has at least one adjacent neighbour which belongs to another region and may have up to a maximum of eight neighbours belonging to other regions (in which case it is an isolated point). The boundary points in a region form one or more boundaries enclosing non-boundary points in the region. A *boundary* is, in general, the set of connected boundary points which is a subset of the set of points comprising a region. The exception is a region which has no non-boundary points in which case all points in the region belong to the boundary. The boundaries of a region can be thought of as the interfaces between the region to which they belong and its adjacent regions, figure 31a.

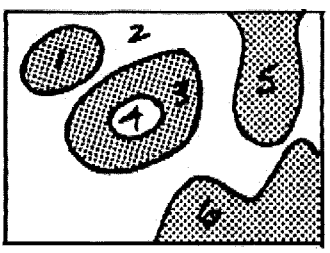


Figure 30 Region map view

Every boundary has one or more complementary boundaries. A complementary boundary is the boundary of an adjacent region which is next to all, or part, of a boundary of the region under consideration, figure 31b.

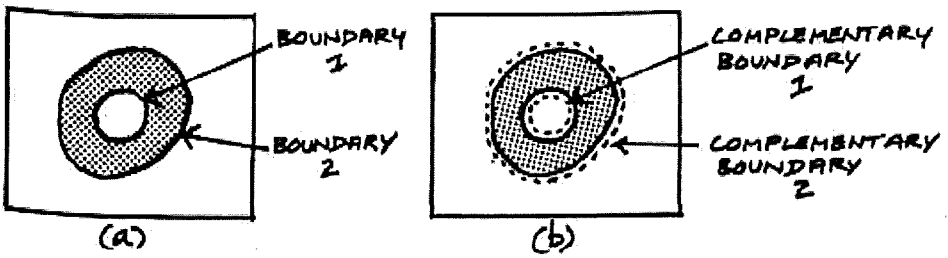


Figure 31 Boundaries and complementary boundaries

In a bi-intensity system the points comprising the complementary boundaries of a region will all have a single intensity value. Thus in a black and white system if a region is black its complementary boundaries will be white.

Two types of region can be identified in terms of boundary and complementary boundary characteristics ; these being referred to as blob and ring regions. A blob, figure 32 is defined as having one boundary and one complementary boundary. When a region is adjacent to the edge of the "region-map" the complementary boundary is viewed as extending across that part of the region under consideration adjacent to the edge of the "region-map", figure 32c.

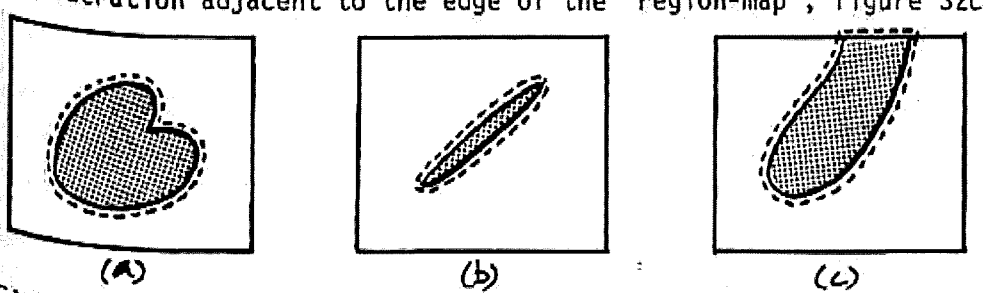


Figure 32 Blob regions

A ring, figure 33, is defined as having one or more boundaries and more than one complementary boundary. Figure 33b illustrates the need for considering complementary boundaries in the definition of ring regions. In general it is sufficient to say that a ring has many boundaries. An exception to this rule is the line lattice

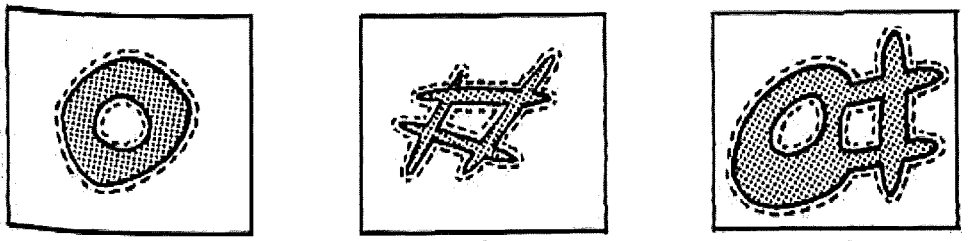


Figure 33 Ring regions

structure illustrated in figure 33b where there is only one boundary. However, although figure 33b only has one boundary it has many complementary boundaries. Algorithms are defined in appendix 1 which extract information about regions and their boundaries and complementary boundaries which can then be used in descriminating blobs and rings.

8.2 FIGURE GROUND RELATIONSHIPS

As stated previously most pictorial art involves visual illusion in the sense that objects depicted are seen as existing in a three dimensional space whilst in fact they are marks on a two dimensional surface. The art of the Dutch artist Escher explores the paradoxical nature of the perception of pictures. In figure 34, Escher not only provides us with a convincing illusion but also reminds us that it is an illusion. In representational art some objects are perceived against the background of others. This is consistent with our understanding of the world in which objects nearer to us occlude parts of those further away. This illusion is not restricted to images which depict recognisable objects. The effect is just as strong in abstract or non-figurative imagery. The English artist, Patrick Heron explores our expectations and creates considerable ambiguity where a shape in isolation appears in front of a surrounding shape at a particular moment but behind it when other shapes are taken into consideration, figure 35. This kind of illusion is often referred to as *figure-ground* since just as in the real world we perceive objects against a backcloth of other objects so a shape (figure) in a picture may appear to be on top of another shape (ground) Figure 35a for example is a surface which is textured and comprises three areas A, B, and C. However, the same picture can be perceived as figure 36b, a black figure with a hole in it above a white ground.

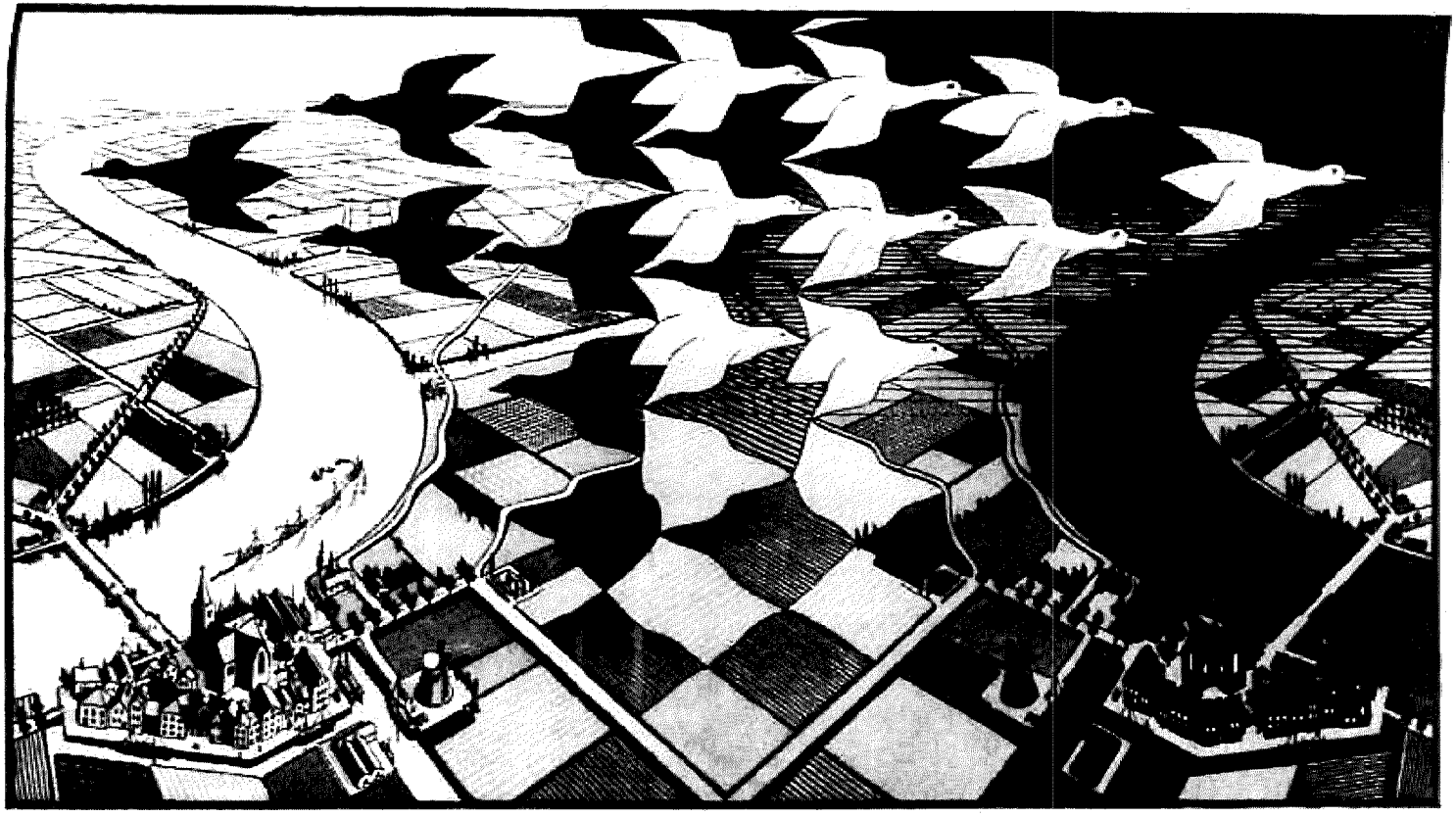


Figure 34

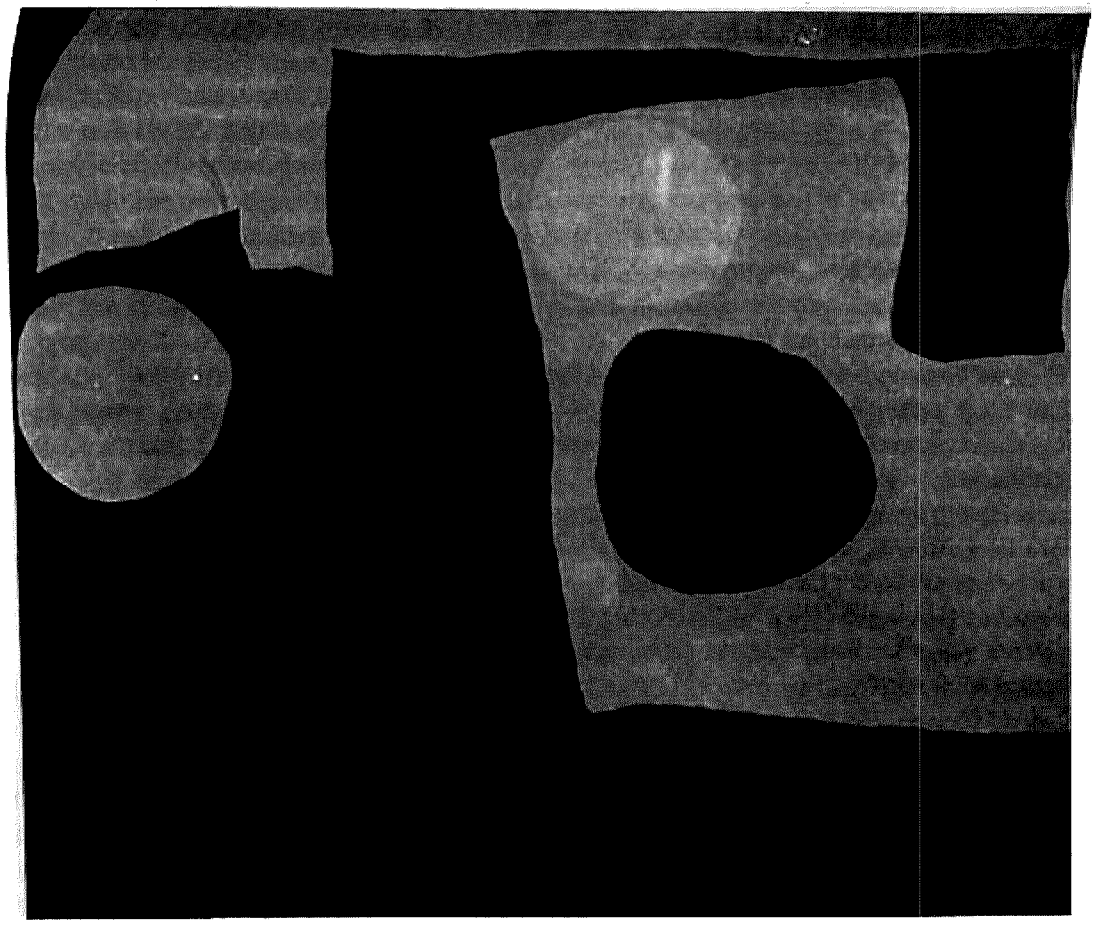


Figure 35

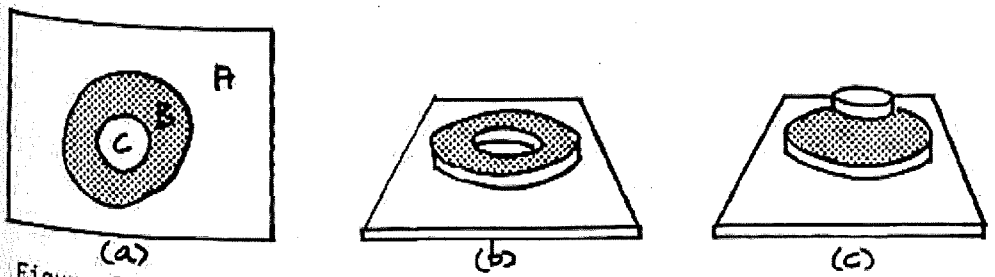


Figure 36 Levels of figure ground

Or as, figure 36c, a white figure on a black ground which is in turn seen as a black figure on a white ground. In figure 36, there are several layers of figure-ground relationships. It is important to recognise that figure-ground structures perceived by the user imply properties which are not properties of the surface, for example depth. Figure-ground then is a subjective property of the viewers perception of the arrangement of shapes on a surface.

8.2.1 EXTRACTING FIGURE-GROUND FROM THE *BITMAP*

As discussed in section 8 the "bitmap" can be viewed as a collection of regions which can be thought of as fitting together like pieces in a jigsaw. The algorithms for extracting regions and boundaries can be viewed as disassembling the "*bitmap*" into its constituant regions. If figure 37 was disassembled then three regions A, B and C would be extracted, figure 37b. If region A was perceived by a viewer as a white figure on a black ground B and if the computer knows which region is perceived as figure and which as ground then the regions implied by the relationship A on B can also be extracted by making suitable adjustments to A and B, as necessary, figure 38. In this particular figure the hole in the middle of region B needs to be filled in since from the figure-ground point of view this portion of the region is merely occluded by the figure A.

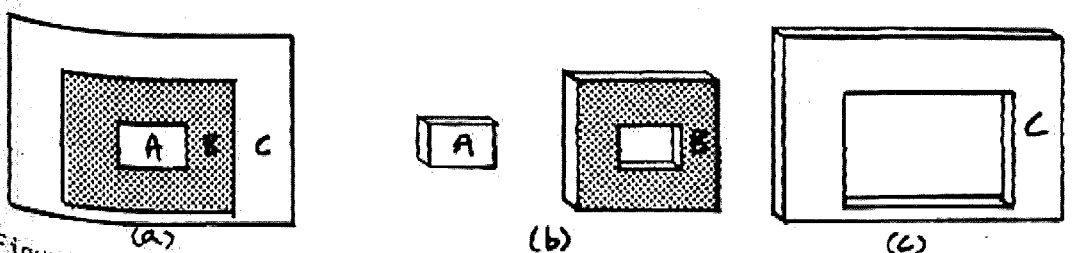


Figure 37 Disassembled regions

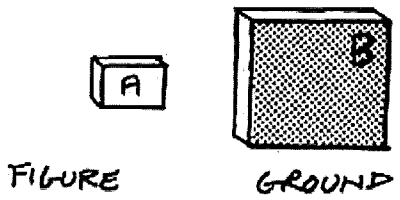


Figure 38 Regions seen a figure and ground

Given certain knowledge about the properties of figure-ground relationships and guided by information provided by the user the computer system can extract structures which are consistent with a figure-ground view of the *bitmap*. To use the words of Micheal Thompson (1972) we can make the computer operate on not only" .... what is there (geometry and engineering) but what we think we see".

8.2.2 SINGLE LEVEL EXTRACTION

As described previously some region configurations can be viewed as having many levels of figure and ground. Thus in figure 39a A can be viewed as a white figure on a black ground B and B as a black figure on a white ground C, figure 39b. Alternatively the picture can be seen as a single level figure-ground where B is a black figure with a hole in it on a white ground A,C, figure 39c.

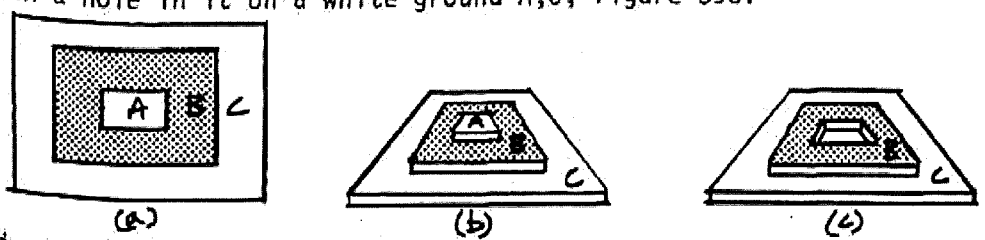


Figure 39 Single level extraction

In the following pages we shall be concerned exclusively with single level figure-ground relationships.

8.2.3 RESTRICTIONS ON FIGURE AND GROUND

An object can be described as a *figure* if and only if it is completely surrounded by points of the opposite tone ; in other words it must be a region. This means that the area marked a in figure 40 does not qualify as a figure, although it might be perceived as a figure, because it is not completely surrounded by B and is therefore part of area A. An object can be described as a ground if and only if it

encloses one or more regions of the opposite tone. As in a figure, a ground must be a region. Thus the class of figure and ground which can be manipulated is restricted and may not always match the viewers perception of figure and ground.

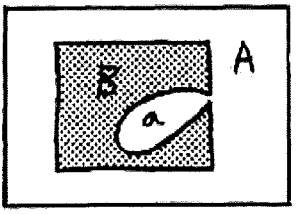


Figure 40 Figure resulting from perceptual closure

8.2.4 EXTRACTING A FIGURE

There are two kinds of figure which can be extracted from the bitmap these being solid, (figure 41a) and holed figures (figure 41b).

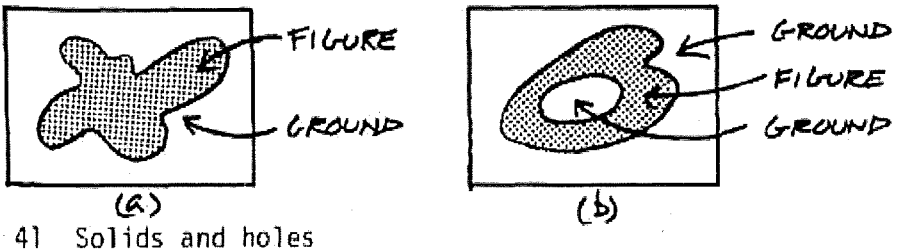


Figure 41 Solids and holes

At the region level a solid figure is a blob region and a figure with holes in it is a ring.

In general, a figure is extracted by reference to a point in the region which is viewed as a figure. Extracting a figure is really a question of extracting a region since no modification needs to be made to the information. The extraction routine retrieves the region identified by a user specified point which is then represented internally as a set of points all of which have the same intensity (i.e. black or white).

8.2.5 EXTRACTING A GROUND

On the other hand extracting a ground from the *bitmap* in general results in modification of the region which is viewed as ground. The reason for this is that a ground usually has something on it,

and enclosed within generally yield a ring. Figures are assumed to occlude those parts of the ground that they cover and thus make the structure consistent with this view interior holes in the ring must be filled. If the point (represented by a cross) in figure 42a was specified as a ground by a in the initial phase of processing the ring illustrated in figure 42b would be extracted.

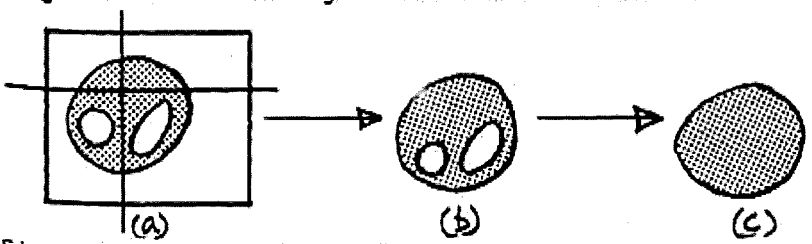


Figure 42 Extracting a ground

Converting it into a ground is a matter of supplying points to fill the interior holes in the ring. In a sense filling in all the figures, figure 42c.

8.2.6 EXTRACTING FIGURE AND GROUND

The previous examples demonstrate the process of extracting figure or ground but it is a simple matter to extract both figure(s) and ground. The point identified by the user to reference the figure(s) and the ground can belcng to either the figure(s) or the ground. Consider, for example, figure 43a, where the reference point is in a figure. Given this point a blob would be extracted which in the context of this operation can only be a figure. Any point on the complementary boundary, figure 43b, can be taken as a reference point for extracting the ground, figure 43c. In the initial phase the perceived ground is extracted as a ring which includes all the regions it encloses any of which can be viewed as figure. The ring is then transformed into a blob that represents the ground. The identified figure and ground are thus obtained.

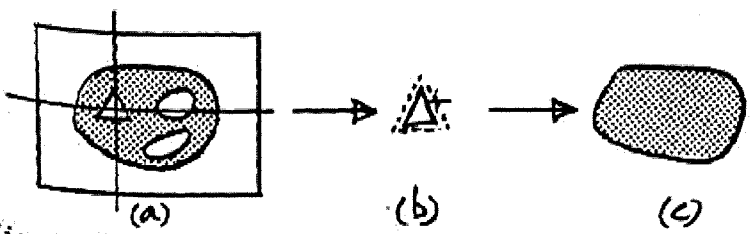


Figure 43 Extracting figure and ground

Extracting figures and ground by reference to a point in the ground is essentially the same operation as figures and ground by reference to a figure and requires no further explanation. In each case the processing results in the set of points representing the figure(s) and ground which (unlike figure or ground extracted directly) contain points of both intensities, black and white.

8.2.7 EXTRACTING FIGURE(S) BY REFERENCE TO THE GROUND

Section 8.2.4. described how a figure can be retrieved from the *bitmap* by reference to a point on the figure. Likewise figures can be extracted by reference to the ground on which they are perceived to lie. Thus the user might request figures on ground where the identified point, figure 44a, is regarded as belonging to the ground. When the extraction process is complete only the figures, or figure are retained, figure 44c. This operation is different to the figure operation described in section 8.2.4 since it provides a way of extracting disjoint figures lying on the same ground.

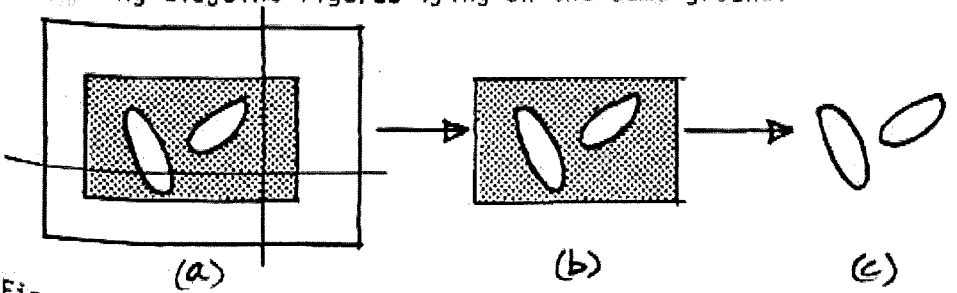


Figure 44

8.2.8 EXTRACTING A GROUND BY REFERENCE TO A FIGURE

This is the complementary operation to that described in the previous section. Here the ground under a figure is retrieved by reference to the figure on that ground.

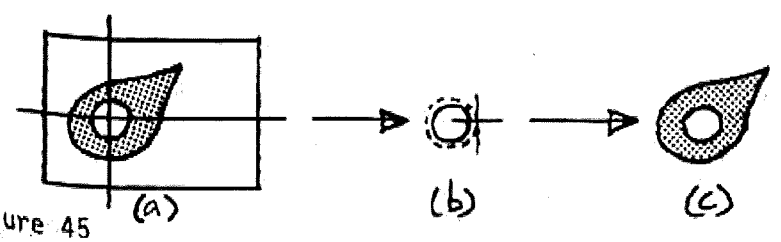


Figure 45

8.3 PERFORMING OPERATIONS ON POINT SETS EXTRACTED FROM THE *BITMAP*

When information is extracted from the *bitmap* it is not labelled as figure, ground or figure(s) and ground. The extraction functions provide the user with a vehicle for communicating his or her perception of figure-ground features and control the generation of information consistent with a figure-ground percept. The result of an extraction function, however, is a set of points which are either all black, all white or a mixture of black and white (as for example the case of a figure and ground). The following sections describe a number of binary and uniary operators which can be applied to point sets generated by the extraction functions described previously.

8.3.1 OVERLAP

From the point of view of image manipulation the OVERLAP operator provides a way of placing one shape on top of another shape, figure 46.

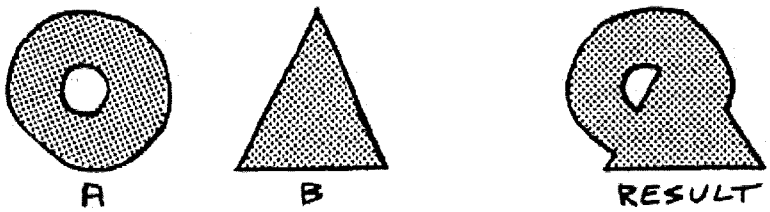


Figure 46 Overlap

This operator is analogous to the set operator union and does result in the union of the sets of points A and B. However, each point has an intensity value which is also taken into account by the overlap operator. Intuitively every point which is a member of both A and B takes the intensity value of A in the new set and all points which are members of one set only (A, or B but not both) pass with their intensity values unchanged into the new set. Figure 47 illustrates the effect of the overlap operator on intensity status in tabular form.

	A	¬A
B	A	B
¬B	A	—

— REMAINS UNCHANGED

Figure 47

The operator merges the two sets A and B such that all points with the same location in each set take the intensity value of the points in A. As stated previously the overlap operator can be viewed as placing A upon B. Alternatively it can be viewed as slipping shape B under shape A. The way the operator is viewed will, of course, depend on the objectives of the user. The sets are merged about the points used by the user to identify the shapes during the extraction phase.

8.3.2 SAME

The SAME operator produces as the new set only those points which are common to both A and B. It is analogous to the set operator disjunction. The effect on intensities values is the same as overlap. Thus the intensity values of the common points in A replace the values of the common points in B, figure 48.

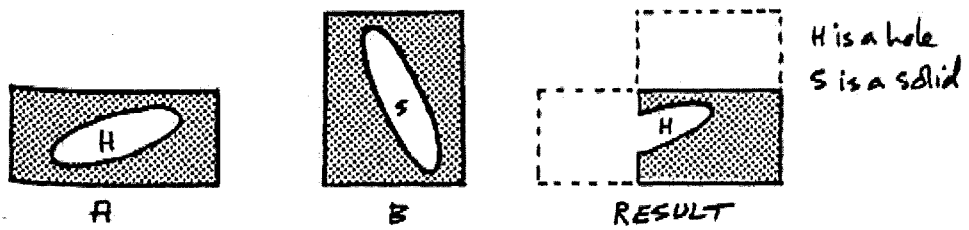


Figure 48 Same

8.3.3 DIFFERENCE

The difference operator produces a new set which excludes common points. Two differencing operators are provided. These being that which differs in A from B and that which differs in both A

and B. Since difference operators work on those points which are different in sets the operators can have no effect on the intensity values of the points in the new set. The difference operators can be characterised by membership functions which return a value of 0 for points not in the new set and 1 for points in the new set. The tables below illustrate the effects of difference operators in terms of the values of the characteristic functions.

DIFFERENCE A,B

	A	¬A
B	0	0
¬B	1	–

– remains unchanged

ALL-DIFFERENCE A,B

	A	¬A
B	0	1
¬B	1	–

– remains unchanged

An example of difference is illustrated in figure 49, whilst all-difference is illustrated in figure 50.

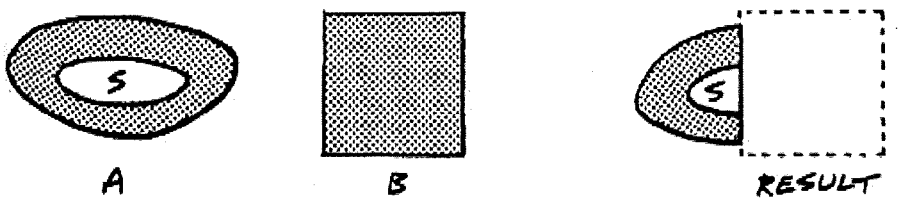


Figure 49 Difference A and B

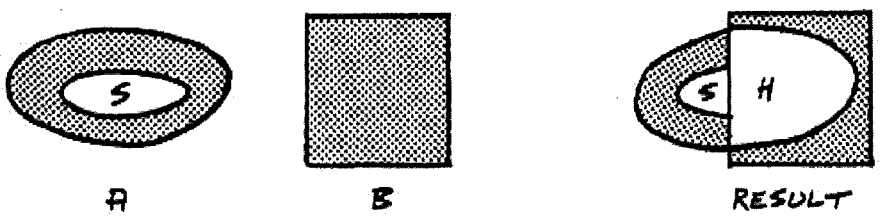


Figure 50 All-difference A and B

8.3.4 INVERT

The invert operator inverts the intensity values of a point set. Thus black points become white and white points are changed to black. The table below illustrates the invert operator where 0 represents white, 1 black, A the set to be inverted and B the inverted set.

A	B
0	1
1	0

8.3.5 DISPLAY

Having extracted and manipulated point sets the display function can be used to display a point set on the screen. The effect of the display operator is to replace equivalent points in the display memory by the intensity value of points. The table below shows the effect of this function where A represents a point in the set to be displayed and B the screen memory.

	B
A	A
¬A	B

8.4 CONCLUSION

In chapter 5 and 6 it was argued that we should expect the artist to be tentative both in terms of the process of making a picture and also interpreting it. Ultimately the effects of changes of mind by the artist will be seen in the changing appearance of the picture that he is working on. The limitations imposed by conventional refresh and storage systems permit the artist little freedom to manipulate pictures in an arbitrary way. The ideas discussed in this chapter set out to explore the possibility of treating the *bitmap* as a map of the screen image from which shapes

can be extracted as they are identified by the user. If it is possible to extract descriptions from the *bitmap* then there is no need to generate shape descriptions at the outset which frequently prescribe implicitly their scope for manipulation. The problem is not straighforward because pictures are perceived to have properties that are not observable in the picture but are constructed in the mind of the beholder. Part of the problem is to identify pictorial concepts that the artist can employ to explain to the computer what it is that he sees. Then the computer, like the brain that manufactures for example a three dimensional world from the marks on a two dimensional surface,must analyse it's "two dimensional surface", the *bitmap* and match the information resulting from this process with the information provided by the user to produce a result that is consistent with what the user sees (e.g. figure/ground relationships).

The figure-ground extraction routines offer a facility which allows the user to identify a class of perceived objects. These objects are not physical properties of the *bitmap*, but they can be understood in terms of physical properties of the *bitmap*, and when identified by the user can provide the information necessary to generate the set of points consistent with the figure-ground percept. Prior to the use of the extraction functions no demand is made of the user to describe to the computer in some way the object he or she is in the act of producing. Even when the extraction processes are invoked they result in a representation not much removed from the *bitmap* which is readily manipulated. All data structures are temporary unless specified by the user and tend to provide a way of remembering pictorial elements rather than a representation of the picture in progress. In essence these facilities reflect an approach which seeks to provide the user with a way of communicating with a computer about pictorial aspects of pictures.

The binary and unary operators provide tools for manipulating information extracted from the *bitmap*. The OVERLAP operator has an obvious and natural correspondence with other physical media. The SAME and DIFFERENCE operators are not analogous to any operation that can be performed using conventional artistic media but they do, intuitively, seem natural in the context of the medium of raster graphics.

Together the extraction and manipulative functions provide the basis of a computer graphics medium which allows the user to operate on shapes as and when they are perceived. This means that the user can keep an open mind both about how he interprets the picture because he is not required to make decisions which constrain future decision making. The user is allowed the luxury of changing his mind about the picture by default because he is not expected, or required, to make up his mind.

CHAPTER 9

THE SYSTEM : IT'S LANGUAGE AND RESOURCES

Eason (1979) argues that designing a computer system for open ended tasks amounts to providing "a repertoire of facilities in the form of a set of resources which can be marshalled to meet a wide range of needs". This chapter is essentially a description of such a "set of resources" which comprise an interactive raster graphics system for artists. The resources for extracting and manipulating shapes described in chapters 7 and 8 that make significant use of the potential of the *bitmap* are, of course, included in the set but since they have been discussed previously are only mentioned briefly here. Implicitly the set of resources provided in a computer system are bound up with the interface language used to manipulate them and its characteristics are of some interest, therefore.

To begin with the strategies for designing man/machine languages as identified by Eason (1979) are examined and the LOGO language is identified as providing a suitable model for the GLIMPS (Graphics Language for the Interactive Manipulation of Perceived Shapes) language.

When talking about computer languages it is usual to distinguish between syntax and semantics. Intuitively the syntax of a language is its structure whilst the semantics is the meaning attached to structural elements and their combinations. In a sense, the semantics of a language tells us what the syntactic elements stand for. The chapter proceeds with what is basically a description of the resources provided by the system in the semantics of the GLIMPS language. In the concluding pages the syntax of the LOGO language is considered with respects to the needs of the artist identified in the third chapter.

9.1 THE INTERFACE LANGUAGE

It is the language, Eason continues, by which the man and computer

communicate that provides the medium by which the user selects, combines and uses available resources. He also stresses that the ability to combine resources in a variety of ways is an important feature of the languages because it enhances the flexibility of the system. This is a subtle point because, assuming that the resources available to the user are sufficient for his needs then his ability to perform a given task will hinge on the organisational and combinational flexibility of the language. For example, if a task requires the combination of resources A, B & C but the language does not allow this combination then the user cannot perform the task. Also, it is the combinational flexibility of a language which makes it possible for the user to personalise a system. If, for example, a user can combine resources A, B & C to form a resource D then he can be said to have personalised the system to meet his own particular needs. However, before pursuing this question any further it is necessary to decide what kind of language is required.

9.1.1 TASK FREQUENCY

A language with the required flexibility is likely to be complex, Eason (1979) suggests that from the designers point of view the simplest solution is a formal command language. In chapter 3 a number of high level programming languages (SPARTA, PICASO) were examined and it was observed that the problem with these languages is that they require the user to acquire considerable knowledge in order to use the facilities they support. Eason (1979) suggests that where the user is likely to make regular use of a computer system a high level language may be appropriate ; but for many users who have "complex open-ended tasks" and are unwilling or unable to devote the time necessary to master a formal language this strategy is inappropriate and may lead to the rejection of the computer system. Clearly, it is less hazardous to regard the artist as an "infrequent" user.

Eason identifies three strategies that are currently being advocated to deal with the problem of the infrequent user : the natural language system ; the human intermediary ; and the evolutionary approach using an adaptive, dedicated interface. The natural

language system attempts to receive and interpret natural language inputs from the user and respond accordingly. Obviously if the user can communicate with a computer as if it were another person the language barrier is dismantled. However there is considerable work to be done before natural language is a feasible choice for the system designer. The use of a human intermediary as an interface between the user and the computer has proved successful in managerial applications and specialist services to the public. Eason observes that although it has obvious attractions it leaves lingering doubts that "the indirect connection between end user and the computer system may rob the user of much of the system's potential". The third strategy takes the view that the final goal could be a formal language but the user should not be taught the language before he starts to work with the system. The central idea is that usage should gradually evolve as the user gains experience of performing tasks using the system. As the users knowledge increases the system adapts by placing more language facilities at his disposal. Thus the system tracks the development of an individual user and is therebye a form of dedicated interface. As Eason notes "the problem is where to start because the user must be able to do something useful with a minimum of initial instruction". There is also the general problem of adaptive systems (discussed in chapter 4) which is concerned with when and how adaptation should take place.

The human intermediary is not a solution to aim for but a compromise between a formal language and a natural language or adaptive system. The intelligent component, implied in the latter solutions, is a human being in the former. On the other hand natural language and adaptive systems are, for the present at least, goals to aim for rather than practical alternatives.

There is, however, a fourth strategy which in principle is the same as the adaptive system without the adaptive function of the computer system. In other words, the goal of this strategy is a formal command language which the user does not have to be taught in detail before using the system. Like the adaptive system the central idea is that the users facility with the language should develop through usage ; a kind of learning by doing. In the

next section an example language will be discussed which it is believed demonstrates the practicality of this strategy.

9.1.2 LOGO - A MODEL LANGUAGE

LOGO is an interpretive language that was developed at the Massachussetts Institute of Technology as a "suitably clear and intelligent programming language" (Papert, 1970) to be used in the LOGO project. Papert (1970) has described LOGO as a "baby LISP", but goes on to argue that it is a "full-fledged" universal language and that its babyish feature is the inclusion of self-contained subsets which can be used to achieve results with very limited instruction.

The aim of the LOGO project was to demonstrate that technology can be used not as machines for processing children, but as something the child can learn to manipulate, extend and to apply to projects. Papert (1970) believed that children learn by doing and by thinking about what they do. McCorduck (1979) points out that perhaps the most controversial part of Papert's program is the idea that children think about thinking. Papert writes (1970), "it is usually considered good practise to give people instruction in their occupational activities. Now, the occupational activities of children are learning, thinking, playing and the like. Yet, we tell them nothing about those things. Instead we tell them about numbers, grammar and the French revolution ; somehow hoping that from this disorder the really important things will emerge all by themselves. And they sometimes do. But the alienation-dropout-drug complex is certainly no less frequent". Papert argues that computation is the richest known source for providing children with better things to do and better ways to think about themselves doing these things. Thus he set out to create an environment in which the child could become highly involved in experiences leading to the growth of intuitions and concepts for dealing with thinking, learning and playing. Papert (1970) reports that LOGO has been used with children of most ages and levels of academic achievement and that in one extended project, within three months, 12 year old children were writing programs to play games, to generate random sentences and even conversational and teaching programs.

Experiments at Edinburgh University (Howe, 1980) also using LOGO, indicate the educational validity of Papert's arguments. One experiment was conducted on 11 to 13 year old boys in a bottom-stream maths class over a period of two years. They spent one hour in a programming classroom while a control group followed the usual time table. At the end of the two years Howe (1980) and his colleagues concluded that "our pupils understanding of mathematics and their ability to do mathematics improved relative to the control group. Also they gained self confidence, becoming more positive in their attitude to maths and much more willing to talk about maths and argue about maths with their teachers".

However, the validity of Paperts arguments is not a matter of dispute here. The reason for referring to this work is primarily because it indicates that LOGO is a language with considerable complexity that has never the less been successfully mastered by children. Of course, children are in the business of learning, as Paperts points out, but they are not necessarily motivated to learn. The evidence suggests that LOGO is not only a language which can be readily learnt but that it also motivates the child to explore the resources it supports in a meaningful way.

9.2 GRAPHICAL EXTENSIONS TO LOGO : THE GLIMPS LANGUAGE

The following section describes a set of graphical functions which have been defined using LOGO as a model. Consequently they are viewed as a set of facili ties which could be implemented as an extension to LOGO. They do not extend the structural capabilities of LOGO but they make it possible for the structural concepts of the language to be employed in the generation and manipulation of raster display pictures.

The reader who is unfamiliar with LOGO will find a summary of the language in appendix 2. Many of its features are explicit and implicit in the discussion that follows. However at the end of the chapter the features of the language that justify the belief that LOGO demonstrates the practicality of a man-computer interface which the user can learn about in parallel with task performance

(ibid 9.1.1) are examined. This has been put off for the time being because it includes examples which use some of the functions about to be described.

These graphical functions provide facilities for the selection and control of graphical devices, real or virtual, and also the extraction and manipulation of shapes perceived in the drawing surface, or screen.

9.2.1 DEVICE SELECTION

A graphical system can provide a number of devices for generating marks. These devices can be either real or virtual. For example a plotter is a real device and the use of device control primitives, such as raising the pen, have specific electro-mechanical effects. A virtual device is one which only exists as a conceptual entity. Its attributes do not necessarily, have any physical correlation. Frequently a virtual device can be implemented using any of a number of physical devices. The drawing devices discussed in this chapter are virtual devices which have been implemented on a raster graphics display. At present there are two drawing devices referred to as the PEN and the ERASER.

The ENABLE function makes it possible for the user to select one, or the other, of these devices. The device enabled becomes the *current device* and all device control functions act on the current device. The selected device is enabled at the origin of the *drawing surface* (the screen) in a raised position (i.e. it is not in contact with the drawing surface). The origin in the case of the screen has a value of zero in both the x and y dimensions which is physically located at the top left hand corner of the screen. The device selected by the use of the ENABLE function remains the current device until it is explicitly disabled by the use of the DISABLE function, or the use of the ENABLE function with a different device name as its argument.

9.2.2 DEVICE ATTRIBUTES

The drawing devices have attributes which are affected by the use

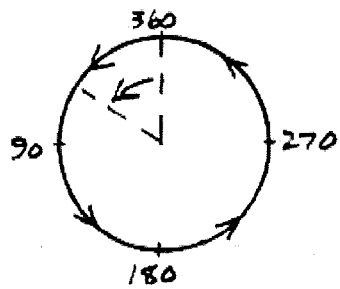
of the device control functions. The language maintains the current state of each device attribute in its workspace and functions are provided which make this information available to other functions.

9.2.2.1 ELEVATION

A drawing device can either be in contact with the drawing surface or not. The *current elevation* status of a device can be modified by the functions UP or DOWN. UP raises the device, and DOWN places the device in contact with the drawing surface.

9.2.2.2 ORIENTATION

A device has a *current orientation* which represents the direction in which the pen is pointing. The orientation of the pen is measured in degrees, in an anti-clockwise direction, from a point due north (where north is the top of the screen).



9.2.2.3 POSITION

A device has a position relative to the origin of the drawing surface. The position of a device at any point in time is described as the *current position* of the device. The current position of a device is defined as x,y coordinate values.

9.2.3 DEVICE CONTROL FUNCTIONS

Device control is exercised by functions which specify displacements of the device relative to its current position and orientation, or locations on the drawing surface. One kind of function tells the device <u>how</u> to proceed and by <u>how</u> much whilst the other tells it

where to go. In examples illustrating the use and effects of these functions the current device is assumed to be the PEN.

9.2.3.1 CONTROLLING WHERE THE PEN GOES

The GO function causes the pen to be moved to the specified position. Thus if the pen is down and the current pen position is 50 on the x axis and 50 on the y axis, figure 51a, and the command GO 100 100 is entered, a line would be drawn between point 50,50 and 100,100, figure 51b.

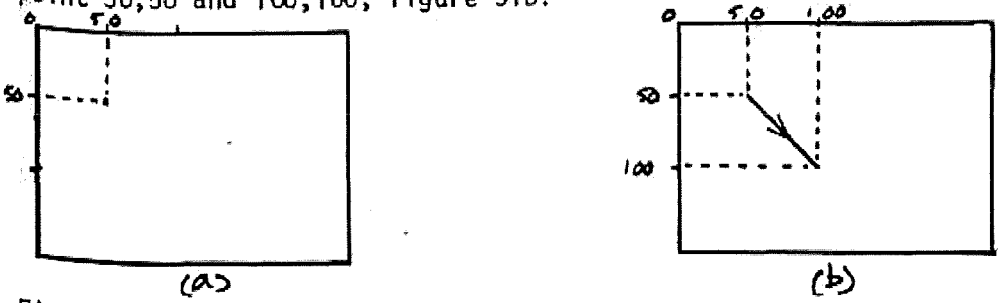


Figure 51

In other words the pen travels between its current position and an absolute position on the drawing surface. Movements of the pen relative to its current position can be performed by first using the BY function. The BY function computes the absolute position derived from its parameter and the current pen position and moves the pen to this position thus the command GO BY 100 100 might have the effect illustrated below, figure 52.

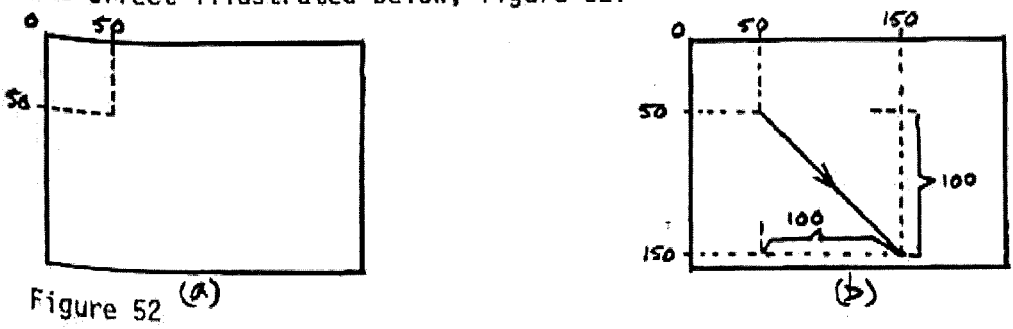


Figure 52

This has the effect of moving the pen relative to the current pen position by the computed displacements in the x and y dimensions. The GO function updates both the current position and the orientation of the device as a result of its use.

9.2.3.2 CONTROLLING HOW THE PEN MOVES

For controlling how the pen moves there are functions which modify its orientation and those which modify its movements in a specific orientation. These are the same functions as those provided by a number of LOGO implementations.

Orientation functions rotate the pen abouts its centre either to the LEFT or the RIGHT of the current orientation. For example, if the orientation of the pen was that illustrated in figure 53a the command RIGHT 90 would have the result illustrated in figure 53b whilst LEFT 90 would have the result illustrated in figure 53c.

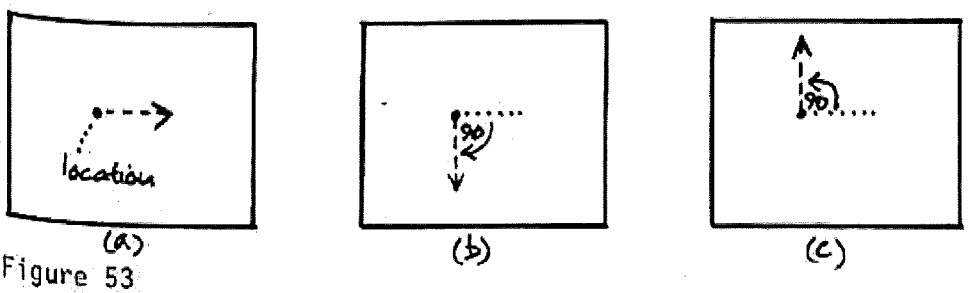


Figure 53

Whilst these commands have a result they have no visible effect. The effect of such a command is only obvious as a result of the execution of one of the movement functions FORWARD or BACKWARD. Consider the following sequence of commands illustrated in figure 54. It will no doubt be obvious by now that the basic syntax of a command is the name of a function followed by a number of arguments. The system is interactive ; the user enters a command terminated by a carriage return which, if syntactically correct, is executed. If a command has an effect (for example a line is drawn) then it will be effected immediately. When a command has been processed the system issues a prompt to the user and waits for the next command. Thus in figure 54, the user enters FORWARD 50 which causes the PEN to move forward along its current orientation 50 units ; this done, the system waits for the next command which is RIGHT 90 and the PEN is rotated by 90 degrees to the right of its current orientation, and so on.

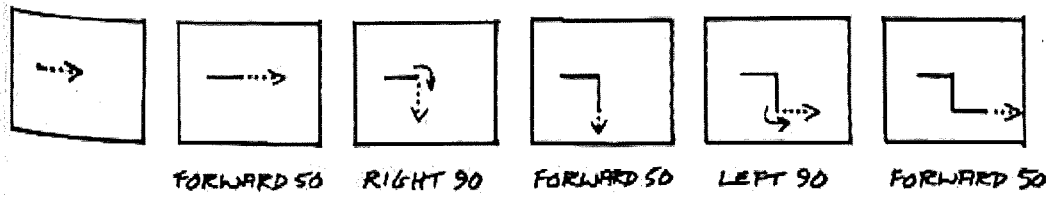


Figure 54

```
FORWARD 50
RIGHT 90
FORWARD 50
LEFT 90
FORWARD 50
```

The functions described in this section characterise drawing in quite a different way to that implied by the functions described in section 9.2.3.1. This difference is largely a question of the manner in which the pen, or drawing device, is controlled. The functions in this section control how the device moves whilst those of the previous section where it goes. The functions which control how the pen is moved seem particularly suited to describing regular polygons. They tend to encourage a geometrical approach to drawing which represents structures in terms of line lengths and angles. This in turn lends itself to a procedural, or modular, approach. Any regular polygon, for example, could be defined as the repetition of a procedure which generates a side and rotates the pen to construct the inner angle at a vertex.

It is certainly more difficult to conceptualise regular polygons using the positional, or coordinate system. To illustrate this, lets consider the problem of defining a square. Using the positional approach the intrinsic structural properties of the square have to be mapped into coordinates. This is not particularly difficult in the case of a square since a mapping procedure can be derived quite easily. However, in the case of a triangle or hexagon the problem is not straightforward. How much easier it is to think of constructing a triangle as 'move forward the side length turn left 120 degrees, move forward etc.'. However there are situations when the positional functions are superior, especially

in instances when parameters are provided by reference to locations on the drawing surface.

Connecting the point A and B in figure 55 by a line is difficult, if not impossible, using movement commands because in this instance what is known is where to go not how. The conventional way of

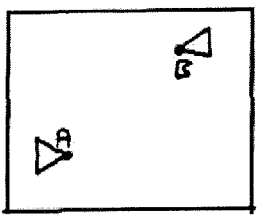


Figure 55

providing these points in an interactive computer graphics system would probably make use of a lightpen, cursor or some other 'locator'. A locator is a device which the user can employ to translate a perceived location into x and y coordinates. This kind of facility is particularly useful in interactive situations because it makes direct use of what the user can see and avoids the necessity of the user providing values directly, which in this context are difficult to determine. Thus A and B could be provided by calls to a function named CURSOR. The line could be constructed using the sequence of commands given below.

```
UP
GO CURSOR
DOWN
GO CURSOR
```

The use of the CURSOR function enables a pair of cross-wires which the user can control to specify a location which is input to the system and becomes the result produced by the CURSOR function. Both kinds of drawing function are therefore necessary since each is superior to the other in performing specific tasks. This example illustrates another feature of the language syntax which is that the arguments for a function can be provided indirectly as a result of evaluating another function. Thus the command GO CURSOR is evaluated in the following way : the GO function cannot be evaluated

immediately because its arguments are not available ; the interpreter then finds the CURSOR function which it can evaluate (CURSOR has no arguments) ; when CURSOR has been completed it produces a result which is the x and y coordinates of the position identified by the user ; these values provide the arguments necessary for evaluating the GO function which can then take place.

9.2.3.3 EXTENDING THE DRAWING FUNCTIONS USING THE BASIC PRIMITIVES

A drawing function which is often included in a graphics system is that which draws a line from one absolute position on the drawing surface to another, e.g. LINE 50 30 60 100. Although this function is not included as a built-in function it is a simple matter to define it as a user function. The language includes a facility which permits the user to effectively, define his own functions. The TO command signals to the interpreter that the user wishes to define a function. The first argument of the TO function represents the name of the new function and subsequent arguments the names of any arguments the new function will use. Having issued the TO command the user can then enter commands preceded by a command number which are not executed immediately but accepted as text. The END function terminates the entry of the text which is then saved, on the users behalf, by the system. The user can then invoke the execution of the saved function (e.g. LINE) in the same way as system functions (e.g. LINE 100 100 200 250). Thus in the example below FROMX and FROMY both take a value of 100, and TOX and TOY, 200 and 250 respectively.

```
e.g.  TO LINE FROMX FROMY TOX TOY
      10 UP
      20 GO FROMX FROMY
      30 DOWN
      40 GO TOX TOY
      50 UP
      END
      LINE 100 100 200 250
```

Similarly symbols can be generated using movement functions.

```
e.g.  TO TRIANGLE SIDE
      10 DOWN
      20 FORWARD SIDE
      30 LEFT 120
      40 FORWARD SIDE
      50 LEFT 120
      60 FORWARD SIDE
      70 LEFT 120
      END
```

The following sequence might then be used to place a triangle at a particular position and orientation.

```
UP
GO 100 100
ALIGN 100 110
TRIANGLE 50
```

The ALIGN function can be used to set the current orientation of the pen. This function is useful since there will often be occasions when the pens current orientation is not known. In the above example after the GO 100 100 command the user will probably not know the current orientation of the PEN (unless he has the foresight to keep a mental record of the effect of previous pen movements on its orientation). However the final effect of the TRIANGLE function depends upon the current orientation of the PEN when it is drawn. The ALIGN function allows the user to set the current orientation of the PEN by aligning it with its current position and the point defined by its arguments, figure 56.

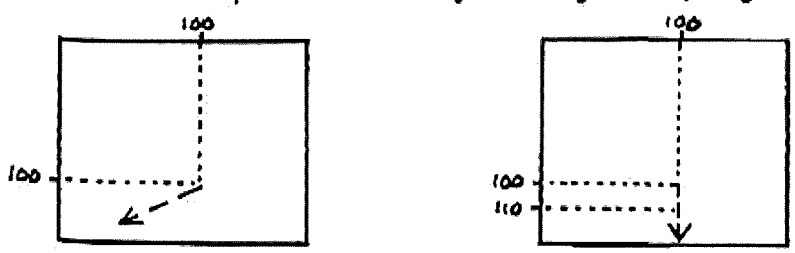


Figure 56

9.2.4 THE ERASER

In the Leicester Polytechnic black and white system the intensity of each point in the screen memory is represented by a 0 or a 1. The precise meaning of these zero's and one's depends on the mode of the display processor which reads the screen memory and displays it on the screen. The display processor can be in one of two modes ; in one mode it treats zero as high intensity (white) and one as low (black), in the other mode one is high and zero is low.

From the point of view of the graphics language the PEN sets any bit in the display memory (drawing surface) which it passes over to 1. Depending on the state of the display processor the one's generated by the pen will either be treated as black or white. Conceptually, however, the pen is best viewed as a device containing an ink of a specified colour which can be used to draw on a surface of another colour.

The ERASER is a device which functionally puts zero's into any bits of screen memory over which it passes. Conceptually it performs the same function as an indian ink rubber. In other words it can be used to erase marks generated by the pen. Once enabled the device control functions described previously can be used to operate the ERASER. The ability to erase marks in this way is a particular feature of the *bitmap* raster display. Put another way, this kind of display makes it possible to draw in different coloured inks. In some contexts, drawing in different coloured ink appears as erasure.

9.2.5. SHAPE EXTRACTION FUNCTIONS

In chapter 7, it is argued that the way to improve facilities by which the user can manipulate what he *sees* in the screen image is to make the computer *see* the image in a similar way. The *bitmap* can be viewed as the computers "screen image" or *visual stimulus* ; the source of its *perceptions*. The problem becomes one of defining processes for extracting *perceptions* from the *bitmap* that are consistent with what the user *sees*. Chapter 8 deals with this problem in the context of figure/ground perceptions.

Extracting shapes from the *bitmap* is made possible by the inclusion of shape extraction functions. The selection of a particular extraction function by the user identifies the kind of shape that is perceived ; essentially there are two types of shape known as figure and ground. The result of the use of an extraction function is, in LOGO terminology, a list. This list is ordered in the following way. The first two words of data represent the *anchor position* of the shape. The anchor position is the point used by the user to identify the shape. The remaining items are ordered in groups of three values each of which provides information about a point in the shape. The first two values represent the location of the point relative to the anchor position whilst the third value represents the intensity of the point, figure 57.

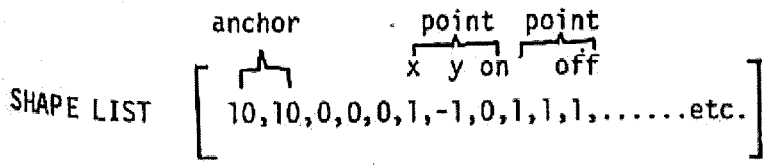


Figure 57

The shape list generated as a result of using an extraction routine will be lost when another function is used later unless it is saved explicitly by the user. LOGO provides a function called MAKE which can be used to save a name and data as an associated pair. Thus the user can save a shape list which can be referenced by its name in subsequent commands.

```
e.g. FIGURE CURSOR
     MAKE SHAPE IT
```

FIGURE is one of a number of extraction functions. Its selection by the user indicates that the shape located at the position identified by its two arguments (which are generated in the above example as the result of evaluating the CURSOR function) is perceived as a figure. MAKE is a LOGO function that can be used to name, and save under that name, data. The result of the execution of a command in LOGO is saved until another command is executed. The function IT makes the saved result from the

previous command available to the next command. In this example a shape list is extracted from the *bitmap* which is then given the name SHAPE in the MAKE command.

9.2.6 SHAPE LIST RELATIONAL OPERATORS

A number of functions are provided which permit specific binary and unary operations to be performed on shape lists. When a binary operation is performed the result is a list which takes the anchor position of the shape which is the first argument in the function call, e.g. OVERLAP SHAPE1 SHAPE2. If the anchor position of SHAPE1 was 10,10 then the list generated as the result of the OVERLAP operation will have an anchor position of 10,10. The general concepts underlying these relational operators and their effects are described in chapter 8.

9.2.7 SHAPE MANIPULATION

Having extracted and operated upon shapes perceived in the *bitmap* the next step may be to display the shape which results. A single function is provided for this purpose which will display a shape at a specified location, e.g. DISPLAY SHAPE 100 100.

As described previously the point in a shape list are defined as displacements relative to the anchor position. When a shape is displayed using the DISPLAY function, the absolute positions of its points, in terms of the screen dimension, are determined by adding the list values to the location defined in the second argument of the function. The operation of the DISPLAY function is the same as the OVERLAP operator except that the second shape in the operation is the *bitmap*. Functions for moving, copying or erasing shapes as described in chapter 7 are not provided as inbuilt functions but they can easily be defined using in-built functions.

```
e.g.  TO COPY FROM TO
      10 FIGURE FROM
      20 DISPLAY IT TO
      END
```

```
TO ERASE AT
10 FIGURE AT
20 INVERT AT
30 DISPLAY IT AT
END

TO MOVE FROM TO
10 FIGURE FROM
20 MAKE SHAPE IT
30 ANCHOR SHAPE
40 MAKE AT IT
50 INVERT SHAPE
60 DISPLAY IT AT
70 DISPLAY SHAPE TO
END
```

INVERT is unary function which inverts the intensity value of each point in the list and ANCHOR returns the anchor position of the specified shape.

9.3 DEFINITION OF FUNCTIONS

The following is a definition of all the functions currently defined for the generation and manipulation of graphical structures and the control of devices. The diagrams used show the syntax of the functions which, in general, the name of the function followed by zero or more arguments. The < > enclose the function arguments which are defined in more detail in section 9.3.7.

9.3.1 DEVICE SELECTION

(a) ENABLE —<device name>—//

e.g. ENABLE PEN

Enables the device identified by the device name by setting all its attributes to default values.

(b) DISABLE —//

e.g. DISABLE

Disables the current device.

9.3.2 DEVICE CONTROL

(a) UP

e.g. UP

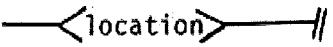
Lifts the current device off the drawing surface.

(b) DOWN

e.g. DOWN

Drops the current device into contact with the drawing surface.

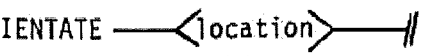
(c) GO <location>



e.g. GO 100 100

Move the current device to the absolute position on the drawing surface specified by location . This function updates the orientation and position attributes of the current device.

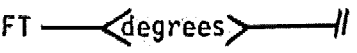
(d) ORIENTATE <location>



e.g. ORIENTATE 50 30

Orientates the current device towards a position on the drawing surface specified by location . This function updates the orientation attribute of the current device.

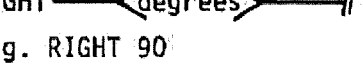
(e) LEFT <degrees>



e.g. LEFT 50

Rotates the current device leftwise about its centre by the degrees specified by degrees . This function updates the current orientation of the current device.

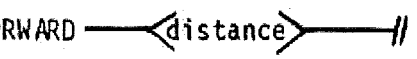
(f) RIGHT <degrees>



e.g. RIGHT 90

Same as (e) but rotation is to the right of current orientation.

(g) FORWARD <distance>



e.g. FORWARD 30

Moves the current device forwards along its current orientation the specified distance .

(h) BACKWARD —<distance>—//

e.g. BACKWARD 70

Same as (g) except that movement is backwards along the current orientation of the current device. Both (g) and (h) modify the current position of the current device.

(i) BY —<relative displacements>—//

e.g. BY 10 -20

Not strictly device control, this function produces a result which is a pair of values representing an absolute position computed using the current pen position and the relative displacements provided as arguments.

9.3.3 DEVICE ATTRIBUTES

(a) ORIENTATION —//

e.g. ORIENTATION

Produces a result which is the angle of orientation, in an anticlockwise direction, of the current device. If a negative value is returned then no device is enabled.

(b) DEVICE —//

e.g. DEVICE

Produces a result which is the name of the current device. NONE is the result if no device is enabled.

(c) POSITION —//

e.g. POSITION

This function returns as its result a pair of values representing the current position of the current device. If the result is negative then no device is enabled.

(d) ELEVATION —//

e.g. ELEVATION

The current elevation status of the current device is returned as the result of this function. If UP a value of 1 is the result, if DOWN a value of 0 is the result. If a negative value is produced as the result then no device is enabled.

9.3.4 CONTROLLING LOCATOR DEVICES

(a) CURSOR ——//

e.g. CURSOR

The cursor function initiates an interaction in which the user controls the movements of a pair of cross-wires ; two lines, one vertical the other horizontal. Upon depressing a special key the location where the 'wires' cross is accepted and the cross-wires are turned off. The result of this function is the final location of the cross-wires when the key is depressed by the user.

(b) LIGHTPEN ——//

e.g. LIGHTPEN

This function performs the same purpose as CURSOR but the physical device which is used to specify the location is a lightpen.

9.3.5 SHAPE EXTRACTION FUNCTIONS

(a) FIGURE —<location>——//

e.g. FIGURE 30 70

The result of this function is the figure identified at the specified location .

(b) GROUND —<location>——//

e.g. GROUND 100 100

The ground identified at the specified location is extracted as the result of this function.

(c) FIGURE-ON-GROUND —<location>——//

e.g. FIGURE-ON-GROUND 50 60

Like (a) this function produces as a result a figure, or collection of figures, but by reference to a location on the figures' ground.

(d) GROUND-UNDER-FIGURE —<location>——//

e.g. GROUND-UNDER-FIGURE 100 100

Like (b) this function produces as its result the ground underneath the figure used to reference it.

(e) FIGURE-AND-GROUND <location>

e.g. FIGURE-AND-GROUND 150 150

Both the figure and the ground identified by the argument location are extracted from the *bitmap* as the result of this function. The location used to identify the perceived structure to be extracted can be either a point on the figure or the ground.

9.3.6 RELATIONAL OPERATORS FOR SHAPES

(a) OVERLAP <shape> <shape>

e.g. OVERLAP TRIANGLE SQUARE

This function overlaps two extracted shapes. The shape identified in the first argument is overlapped onto the shape identified by the second argument.

(b) DIFFERENCE <shape> <shape>

e.g. DIFFERENCE CIRCLE SQUARE

This function produces as its result that part of the shape specified in the first argument which is different to the shape specified in the second argument.

(c) ALL-DIFFERENT <shape> <shape>

e.g. ALL-DIFFERENT OBLONG POLYGON

This function produces as its result all points which are not common to both shapes.

(d) SAME <shape> <shape>

e.g. SAME OBLONG CIRCLE

All the points common to both shapes are produced as the result of this function.

(e) INVERT <shape>

e.g. INVERT POLYGON

The intensity value of all the points comprising the specified shape are inverted as the result of this function.

The precise effects of each of the above operators is described in chapter 8.

9.3.7 SHAPE MANIPULATION

(a) DISPLAY —<shape>—<location>—//

e.g. DISPLAY CIRCLE 100 150

This function displays a shape at a specified location.

(b) FILL —<location>—//

e.g. FILL 100 100

This function fills in a black or white region which is completely bounded by points of the opposite tone.

(c) ANCHOR —<shape>—//

e.g. ANCHOR SQUARE

This function returns the anchor position of the specified shape.

(d) GETPIXEL —<location>—//

e.g. GETPIXEL 100 200

This function returns the value of the pixel located at the specified location in the *bitmap*.

(e) PUTPIXEL —<value>—<location>—//

e.g. PUTPIXEL 0 150 12

This function puts a value (which must be 0 or 1) into the specified location.

9.3.8 DEFINITION OF PARAMETERS

As in LOGO any graphical function parameter can be provided directly or indirectly as the result of a function :

e.g. GO 100 100
or GO CURSOR

This section defines the meaning of parameters defined directly.

(a) <device name>

e.g. ERASER

Device name is a 'word' of data in LOGO terminology. This word specifies a particular device supported by the system.

(b) <degrees>

e.g. 30

A numeric word in the range 1 to 360.

(c) <location>

e.g. 125 8

Two numeric words, the first of which specifies an x coordinate, the second a y coordinate within the bounds of the drawing surface.

(d) <distance>

e.g. 89

A numeric word.

(e) <shape>

e.g. 10,10,0,0,1,1,0,1,1,1,1

A shape is a list in LOGO terminology.

(f) <relative displacements>

e.g. 12 -15

Two numeric words, the first of which specifies a displacement in the x axis, the second a displacement in the y axis.

(g) <value>

e.g. 1

A numeric word.

9.4 IMPLEMENTATION USING GOLL

GOLL is a language designed, and implemented at Leicester Polytechnic, as a computer science final year degree project (Bowden, et. al. 1979). It is based on the ideas propounded in this thesis and is LOGO like in appearance. It includes a facility which permit the user to define function. For simplicity, a subset of GLIMPS has been implemented as an extension of GOLL. In many cases, this was achieved by defining user functions. However, since GOLL contains non of the image extraction facilities described in chapters 7 and 8, in those cases it proves necessary to extend GOLL (Scrivener, Schappo, 1981).

9.5 LEARNING BY DOING

Earlier it was suggested that, from the system designers point of view, it is safer to regard the artist as an "infrequent" user (section 9.1.1). Eason's notion of an "infrequent" user implies the characteristic of resistance to too much learning by instruction. Thus, viewing the artist as an "infrequent" user is less likely to lead to the system designer making gross assumptions about his ability or desire to learn how to use a computer system. However, there are good reasons for taking this view other than mere caution.

In general, an artist learns about a new medium of expression by working with it ; in Papert's terms, he learns by doing. The use of a medium is not usually preceded by extensive theoretical instruction. Thus, for example, a knowledge of the theory of mixing colours is not a pre-requisite for putting paint to canvas. The use of most artistic media involves an element of craft ; a combination of intellectual and physical control. And, in practise, it would seem that there is no better way of acquiring manual skills than practise. Thus the old adage "practise makes perfect" has a distinct ring of truth about it. The importance of practise was certainly recognised in C18 and C19 academy schools where the art student was required to practise for hours his drawing skills in the copying and life rooms, before being permitted to progress to the difficulties of painting and colour. Although contemporary artistic training is perhaps less rigorous in this respect, and students are permitted to develop skills in a more personal manner, the method of learning, in principle, has not changed. Thus it is "natural" for the artist to learn how to use a medium by using it. Similarly the artist tends to learn about the possibilities of a medium as a result of discovery during practise, not by instruction. Learning the mechanics of using, or controlling a medium can be compared to learning an interface language in the context of computer systems. They are both, in essence, processes of learning how to control resources. In general, the processes of learning how to control facilities and how they can be utilised are not distinct activities for the artist. Instead they tend to take place in parallel ; each encouraging development of the other.

In this respect learning about a new medium is rather like learning to drive a car. It is not usual for a person to be taught to drive by extensive instruction in the purpose and effects of the controls and likely traffic situations. There comes a point when instruction is no substitute for practise ; when the learner driver must be permitted to experience the effects of controlling a car for himself. The learner driver develops a facility in handling the controls in parallel with organising them to perform tasks (e.g. making a right turn). Similarly, in the context of art, learning about the mechanisms for controlling available resources takes place in conjunction with learning to manipulate resources to perform tasks.

It is therefore, unnatural to consider separating learning about the interface from learning about the facilities provided by a computer graphics system when the artist is the proposed user. Furthermore, in either case of interface or resources, learning by doing is preferable to learning by instructions.

9.6 *EASE OF USE*

An examination of computing literature quickly reveals that the term "ease of use" has no hard, or precise meaning. It is a fuzzy term that can be applied to all techniques that can be shown to ease the users difficulty in communicating with a computer. Often a users difficulties arise as a result of a lack of specific skills or unfamiliarity with a mode of communication. In the art context, for example, many artists are not competent typists, and since one of the most general input devices to a computer is the keyboard difficulties can arise if it is the principal input mechanism. Often, where a keyboard is used, attempts are made to reduce the actual number of key depressions required to input a command. The use of numeric codes in ART-1 would seem to be such an attempt. It can be argued that this kind of technique makes a system easier to use because, by taking into account the users lack of skill in typing, it reduces his difficulty in entering commands. Critics might argue that codes of the kind used in ART-1 are unnatural and consequently difficult to remember which can, in turn, make the system difficult to use because the

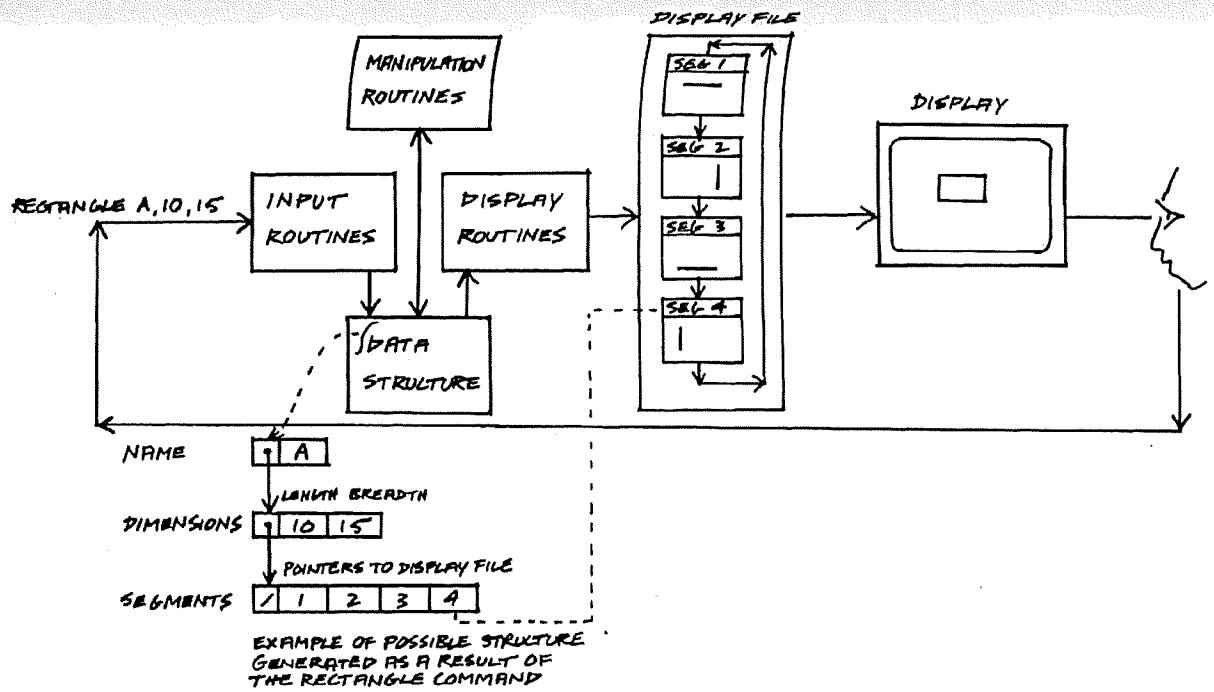


Figure 22 Typical vector graphics system

user will be prone to error due to failure of memory. Thus the argument for the more natural command language of PLAD asserts itself ; a system is easier to use if its command language is easily memorised. A contrary argument is that commands in a more natural language, of the kind used in PLAD, can take a frustratingly long time to enter and because of their length are more likely to be subject to key depression errors.

This discussion illustrates two points : the first is that there is some dispute amongst system designers and researchers as to what it is that makes a system *easy to use* ; and secondly that *ease of use* can be employed to refer to techniques that assist various aspects of the user (e.g. physical skills, memory).

In the following pages the *ease of use* of the language will be considered in terms of the following questions : does the language permit learning by doing? (i.e. can the language be learnt progressively?) ; can the user personalise the system by adapting it? (i.e. can the user combine resources?).

9.6.1 EASE OF LEARNING BY DOING

An important feature of the language, as far as learning it is concerned, is the simplicity of the command syntax which is, in general, a function name followed by a number of arguments. Having mastered the syntax of one command the user has, in effect, mastered all commands. This is not true of many high level languages where the syntax of different statements often have little similarity. One of the difficulties of learning FORTRAN, for example, is its syntactic complexity. Furthermore the utility of statements in many high level languages only becomes obvious when combined with other statements, For example, the DO statement in FORTRAN has very little obvious utility on its own and it is only when it is combined with other statements that its usefulness is realised. In this sense then, the DO statement is not independently useful. Consequently it is the case that the DO statement must be combined with other statements by the user to perform a specific task. This is characteristic of many high level languages and encourages a wholistic approach that tends to separate the process of learning

a language from using it to perform tasks.

On the other hand, many of the commands in LOGO have a utility which does not necessarily rely upon other commands and therefore can be employed by the user without reference to, or knowledge of, other commands. In this sense they are independent of each other. It is the relative independence of commands and the simplicity of the command syntax which make it possible for the user to employ subsets of the language to achieve results with the minimum of instruction. For example, the user might be given as an initial subset the commands : ENABLE, UP, DOWN, RIGHT, LEFT, FORWARD and BACKWARD. If the user was instructed in their use with the PEN as the device enabled this subset would provide a simple drawing facility. Instruction in the notion of the ERASER and the introduction of the DISABLE command would make additional resources available to the user. Later as the user develops an understanding of one subset then new subsets can be introduced. Thus by the gradual introduction of command subsets the users knowledge of the language develops in conjunction with knowledge of the resources it supports. In other words the language, in its structure and syntax makes it possible for the user to learn about both the language and use the resources it supports at the same time. Appendix 3 is the record of the authors introduction to the Edinburgh University LOGO system that illustrates a process of learning by doing.

9.6.2 EASE OF ADAPTATION

There are a number of ways in which we can talk about adaptation in the context of a man-computer system. For example, the computer can adapt in response to the man, or the man can adapt the computer. Eason's third strategy for dealing with the inexperienced user is concerned with the former kind of adaption. No attempt has been made in the system to provide dynamic adaptation in response to the user. However, the system does provide a passive substitute in the sense that the structure of the language is such that the user can learn it as he goes along. In practice, the users success in learning the language in this way depends upon good documentation and instruction manuals. In this section the power of the language

with respect to the second kind of adaptation mentioned (the man adapts the machine) will be considered.

The principal mechanism by which the user can adapt the system is the user defined function (UDF). This facility allows the user to encapsulate his own functions in UDF's. In this way he extends or adapts the language and gives it a more personal meaning. If he wishes the user can define his own set of graphical primitives (e.g. square, triangle). Providing a facility that allows the user to define his own set of graphical primitives is not equivalent to providing them as inbuilt functions even if the two primitive sets which result were the same. Eason argues (1979) that the most likely adaptive component in a task performing system is the user and that if he is to be adaptive he must have the discretion to interpret a task and select a method of execution. He also points out that users may model the task in different ways. An interesting illustration of this problem comes from Herot (1974) who describes how a program for latching sketched lines, input to a computer using a tablet worked better for some people's sketches than for others. "It appeared", Herot explains, "that the programmer had embedded a particular model of human sketching behaviour that fits some users more closely than others".

Providing graphical promitives such as a square or a triangle in a system is to assume that the way they are generated is not going to be important to the user. However, it is often the case that the way a task is performed is at least as important as the goal of the task. It is very easy to ignore this fact when designing a computer system and to treat facilities as equivalent because they perform the same function. Consider for example the problem of providing a facility for moving a graphical entity from one place to another. In some graphic systems this can be done by pointing at the object to be moved and then pointing at its new position. The object disappears from its old position and then reappears at its new location. Some systems allow the user to "drag" an object from one place to another. This, as the name suggests, is rather like dragging a piece of card around by placing your finger on it.. When the user is happy with the new location of the object being dragged he signals this fact to the system and terminates the

process. Both method has the same function but the way that function is performed is quite different in each case. Furthermore, depending on the context in which the move takes place one method may be superior to the other. Dragging, for example, is the better method for positioning a shape relative to other shapes because it provides feedback about the position of the shape during the process which the user can utilise to locate it exactly. From the users standpoint then, the way a task is performed may be important in terms of his objectives.

The UDF is the users way of modifying the system to include function the method of execution of which are at his discretion. Below, two methods are defined for drawing a square are illustrated :-

```
TO  SQUARE
10  FORWARD 50
20  LEFT 90
30  FORWARD 50
40  LEFT 90
50  FORWARD 50
60  LEFT 90
70  FORWARD 50
END

TO  SQUARE
10  GO 100 100
20  GO 150 100
30  GO 150 50
40  GO 100 50
50  GO 100 100
END
```

Each performs the same function of drawing a square but the methods used have different implications in use. To include one method, in preference to the other, in a system is to assume that it is the more general way of modelling the task. The view taken in this thesis is that it is better to avoid this kind of assumption (which is likely to be wrong in any case) and instead provide facilities which allow the user to model his own preferences (whether they take

the form of graphical entities or not) in his own way. The aim has been to avoid, as far as possible, the problem identified by Herot of embedding in the system "a particular model of human sketching behaviour that fits some users more closely than others". The way to avoid this, it is believed, is to provide facilities for combining and organising the resources of a system. The UDF represents the principal mechanism for combining resources and therebye adapting the system.

9.6.3 DEVISING PLANS

The UDF is also a way of defining plans and as such performs a function in respect of the issues discussed in chapter 5. The UDF provides the mechanism for *pre-planning*. Two kinds of planning were identified in chapter 5 described as event and goal orientated. Both types of planning are possible within the language. LOGO contains conditional functions which can be used in event orientated procedures. The example below makes use of the IF...THEN and IF...THEN...ELSE functions.

```
TO  AHEAD VALUE
10  UP
20  FORWARD 1
30  GETPIXEL POSITION
40  MAKE VALUE IT
50  BACKWARD 1
60  DOWN
END

TO  TURN COUNT
10  LEFT 90
20  SUM COUNT 1
30  MAKE COUNT IT
END
```

```
TO  PROCEED
10  AHEAD VALUE
20  IF VALUE=1 THEN TURN COUNT ELSE FORWARD 1
30  IF COUNT=50 THEN STOP
40  GOTO 10
END
```

The PROCEED function is defined by the user to move one unit (pixel, bit) forward unless the position AHEAD is already black in which case the pen is rotated to the left. Every time the pen is rotated the numeric word COUNT is incremented by one (lines 20, 30 of the function TURN). When the number of turns undertaken equals 50 the execution of the PROCEED function is terminated by the STOP function. Thus it can be seen that PROCEED responds to events arising as it is executed. In this case the events relate to the state of the *bitmap* during the execution of the procedure. Examples of goal orientated procedures have already been provided in those illustrating the definition of a square.

9.7 CONCLUSION

In chapter 3 it was argued that the systems reviewed do not adequately resolve the difficulties that can arise as a result of the often conflicting demands of the *ease of use* and *usefulness* of a system. In this chapter the emphasis of *ease of use* is not placed on the ergonomic issues (important as they are) but on learning how to control resources and the flexibility of the language in terms of modelling tasks. It has been argued that, typically, the artist learns how to control the resources of a medium in parallel with task performance. Skill in using a medium and the comprehension of its potential for realising artistic objectives develop in unison. Furthermore, it is proposed that the business of learning in the artistic context is very much one of learning by doing. The "ease of use" of the system discussed here has been largely concerned with the question of learning a language in parallel with using it to perform tasks. LOGO, it is argued, provides a structure which makes it possible for the user to perform tasks with a minimum of initial training using subsets of the language. Also it is believed that the knowledge of the language can be developed progressively by

the introduction of new subsets as the user's knowledge of the resources increases.

Another aspect of "ease of use" discussed here relates to how easy it is for the user to perform tasks using the GLIMPS. There is no simple answer to this question although an adequate solution may be fundamental to the *usefulness* of a system. No matter how potentially useful the resources of a system are if the language for combining them does not allow the user to perform tasks in a way which is natural to him then its usefulness is limited. The problems are that we cannot predict in any precise way what tasks the user will want to perform. Also users may model the same task in different ways and a particular users model might be easy for him but difficult for other users or visa versa. Determining a "universal" task performing strategy that might be embedded in the interface is perhaps impossible. It is more useful to rephrase the question and ask whether GLIMPS is flexible enough to model the same task in a variety of ways. It has been argued that it is and that in particular the user defined function facility comprises the primary mechanism by which the user can adapt the language to suit his objectives.

In conclusion it is felt that the GLIMPS language described in this chapter is a step towards creating a balance between *ease of use* and *usefulness*. These terms have been considered here with respect to the syntax of GLIMPS rather than its semantics ; to its structure rather than the resources it supports. The structure of GLIMPS provides *ease of use* because it can be learnt progressively in conjunction with performing tasks ; it provides facilities that allow the user to adapt the language by effectively adding new, semantics, or resources ; it has sufficient redundancy to permit tasks to be performed in a variety of ways. These things contribute to the language as a medium by which the user can, in Eason's terms, combine resources in a variety of ways. There is, of course, the *ease of use* and the *usefulness* of the resources to be considered and in the concluding chapter we will return to this question.

CHAPTER 10

CONCLUSIONS

One of the advantages of writing a thesis is that it provides an opportunity to stand back and see one's work in some kind of perspective. It is often difficult, as everyone knows, to see an overall pattern in one's work when particular areas are being examined in isolation. It is now time, in this the concluding chapter, to take stock of what has been achieved and what remains to be done.

In retrospect, it appears that the issues tackled in the project fall broadly into those concerned with the task performing language and those dealing with the kind of image generative and manipulative resources that should be provided. Although problems of the interface language and picture making resources are interconnected, for the sake of clarity they will be discussed independantly in the following pages. Indeed it is not only convenient to do this but also helpful because it helps to determine what problems, if any, are particular to each area of study.

10.1 THE TASK PERFORMING LANGUAGE

In a computer graphics system the artist cannot get his hands on the picture making resources in as direct a way as that possible using conventional media. Instead he must pass messages to the computer over a route which he can never travel.

This distancing of the resources from the artist can cause the computer to appear a strange kind of creative medium. The language the artist uses to communicate his intentions to the computer, it has been argued, should be such that it does not make the distance between the man and the resources of the computer appear impossible to traverse. The *ease of use* of the task performing language is, in this context, a measure of its success in achieving the forementioned objective.

The fact that one has to "talk" to a computer in order to utilise the resources it provides is often seen as a disadvantage. The interface between the man and the computer can take on the appearance of a barrier that must somehow be overcome and leads to a view of the task performing language as a thing that cannot be avoided but should at all costs be disguised. From the outset, then, one tends to take a negative view of the need to describe, or communicate, one's objectives to the computer.

It is easy to think of applications where having to "talk" to the computer can be a nuisance and where as a consequence it can appear as a barrier. Evans (1972) system for collecting information from patients concerning their ailments is one example. As the objective of the system is simply to collect information, anything that stands in the way of this goal is a problem. In this kind of situation the computer is being used not to augment or extend mans ability to perform certain tasks but provides a means of automating some of his functions. Thus in Evans' system the computer collects the type of data that a doctor would normally have to extract from a patient in order to make a preliminary diagnosis. The point is that the computer provides a means to an end and as a means it is of no special interest. In this context the task performing language can be seen as a negative factor that must be neutralised.

However to take too negative view of the task performing language is to ignore the possibility that having to communicate with a computer may have a positive side to it. The positive aspect is implicit in Paperts philosophy. In writing of the impact of the computer and its effects on teaching methods he says; "The most important (and surely controversial) component of this impact (i.e. the computers') is on the child's ability to articulate the workings of his own mind and particularly the interaction between himself and reality in the course of learning and thinking." (Papert, 1972). When devising a program to perform some task one is to some extent elucidating one's objectives. Often the process becomes one of externalising in the form of a computer programs one's thought processes ; one's way of doing things. The need to "explain" things to a computer ; to issue it with instructions can encourage, in a positive fashion, thinking about

one's way of performing tasks.

The selection of LOGO as a model for the interface language whilst arbitrary in one sense (there are many similar computer languages) was based upon a belief that the act of programming helps the user to marshall his thoughts and organise his thinking. In its definition GLIMPS has structural features that are common to most programming languages ; in particular the ability to define sub-processes (U.D.F's). As Eason (1979) observes an important feature of a task performing language is the ability to combine in a complex way the resources it supports. The features that make this possible may be thought of as the resources of the language as distinct from the resources it supports (e.g. those for image generation and manipulation).

If we consider for a moment the medium of painting it can be shown that it permits two basic generative/manipulative functions. Paint can be applied to a surface or it can be removed from a surface. The history of the medium of painting reveals the invention of numerous techniques for the application and removal of paint that can be used in subtle and complex ways to perform tasks. Thus they might be used by an artist to achieve a MOVE. For example an artist might wish to move a circular shape in a picture to another position. This he could achieve by painting or scraping out the shape and repainting it in its desired position. Each artists develops a preference for particular methods of painting and scraping that become automatic and contribute to his style. Similarly the ability to define functions using GLIMPS provides a way of combining "painting" and "scraping" to achieve objectives. Thus it offers the artist a means to automate his working processes. It is not the automation in itself that is regarded here as important but the process of thinking out the automation that promotes an examination of the nature of method.

The "user defined function" is a language resource that encourages the specification of functions that encapsulate the kind of high level process (e.g. move) that the artist take for granted and uses automatically when working with the medium of paint. The UDF is a resource of the language and with other language resources

contributes to its *usefulness* as distinct from the usefulness of the imaging resources of the system. This *usefulness* resides in the power of the language as a vehicle for defining picture making procedures. The act of specifying such procedures, it is felt, promotes thinking about them which is beneficial.

It is recognised that the last statement is a question of belief. There is little evidence, other than the work of Papert, that supports the view that the actof devising programs for a computer does encourage thinking and learning concerning the concepts being manipulated. However, if there is any truth in the assertion then it points to a positive and rewarding aspect of communicating with machines. Clearly, there is work to be done to determine first of all, whether the hypothesis has any foundation and secondly, if the answer is affirmative, what language features (e.g. process control) are likely to be of use to the artist. For example, one can imagine that the ability to backtrack or compare solutions might be functions that could be supported by language resources. Finally, there is a degree of speculation involved in all this in the sense that it is implied that doing things differently (e.g. instructing a computer) can be beneficial and this speculative aspect may also be observed in the imagined resources. For example, backtracking is not a facility of painting. Thus one might propose that the work ahead should not be restricted to devising computer facilities for doing the same kind things as, lets say painting, but that it should be looking to discover ways of doing things differently.

Earlier it was suggested that the distancing of the artist from the image making resources is responsible for some of the strangeness of the computer as a creative medium. If one puts to one side the question of whether or not having to"talk"to a computer is advantageous there remains the question of how the user actually enters instructions. Does he type instructions in at a keyboard or point to items on a menu list? Clearly the way instructions are issued is going to influence the *ease of use* of the system. In the event this question has hardly been touched upon in this thesis. Quite nautrally, as one works certain problems take on particular significance and one finds oneself focusing on them to the exclusion of others. In fact there are well established techniques for

interactive input and control; a whole catalogue. However, they fail to answer the real questions? The way commands are issued is only a means to end ; this being to manipulate the resources supported by the task performing language. As such the means will only appear as good as the end. In order therefore to make some comparative study of command input techniques it was felt that the most pressing problem was to improve the computer facilities for image generation and manipulation. In consequence, some of the criticism levelled at earlier systems for artists have not been answered in GLIMPS. Considerable work remains to be done both on the character of the language and the way in which commands are issued using it.

10.2 THE PICTURE GENERATIVE AND MANIPULATIVE RESOURCES

In retrospect, it now seems obvious that the crux of the project has been concerned with problems of image generation and manipulation ; in particular certain questions regarding image manipulation (and manipulation as a generative process) have taken on paramount importance.

The medium of painting provides a number of resources. It provides implements for applying paint to a surface and for removing it from it. There is also of course the paint itself. These are its physical resources. Coupled with these are the actions that they permit, for example applying paint (paint) and removing it (erase). These are also resources in the sense that they are actions permitted by the physical characteristics of the medium. As we have seen the artist organises these primitive actions to perform complex tasks. There are in fact many kinds of erasure and painting possible due to the variety of painting implements at the artists disposal.

As far as this project has been concerned the kind of implements for generating and manipulating pictures that can be provided in a computer system has not been explored. Indeed any number of conventional implements such as a tablet have not been included. This is not to say that the question of what kind of implement could be provided is not a valid one or that the devices excluded from GLIMPS were omitted deliberately. The problem of research is not

necessarily to provide fully functioning systems. Rather its role is to demonstrate that certain things are possible and to show how. It remains for others to take up the research work and develop it, for example, in a working applications system. The issue of computer graphics implements is worthy of research. It has not been undertaken in this project simply because other issues have and some implements have been ignored because they were not felt to be necessary to the research.

The research has hinged around the kind of actions the artist might perform. In particular it has considered what higher level actions (e.g. FILL) could be provided to enhance or assist the artist in the performance of tasks. In painting, although the actions performed by the artist are of two basic kinds he can apply them to any part of the picture at any time. As has been pointed out previously this is not true of conventional computer graphics where the artist only has access to those parts of the picture which have been given a structure. Furthermore operations on the picture are always in terms of its given structure. It has been argued that this both demands that the artist must make up his mind from the outset and also that it restricts future changes of mind significantly. It has been argued that for a number of reasons these constraints are unsatisfactory from the artists point of view.

Figure 58 shows the conventional view of computer graphics. The

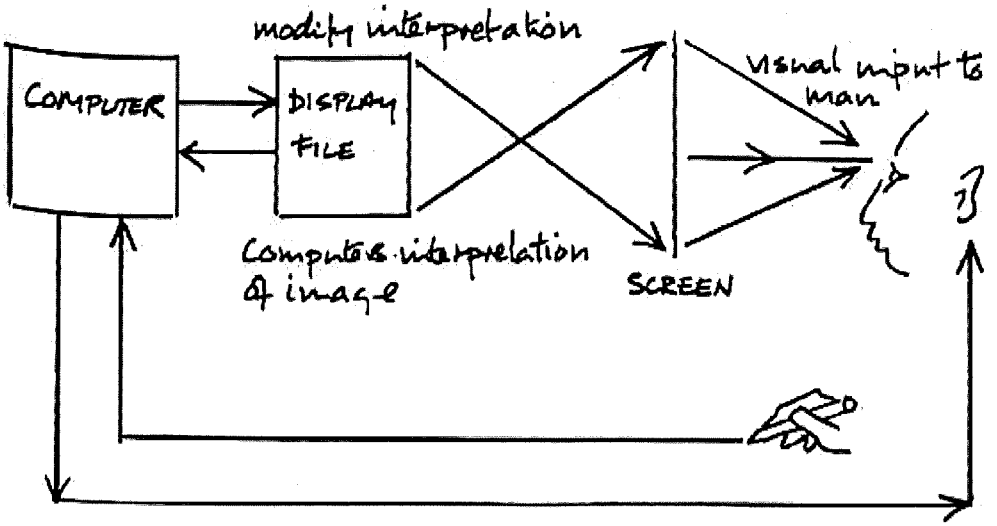


Figure 58 Communicating about an interpretation.

Visual input to the user comes from the screen and it is this that the artist interprets. The pattern on the screen is produced by processing the what has been described by the author as an interpretation of the pattern. When the user initiates an action affecting the pattern he sees, he sets in motion modifications or changes of computers interpretation of the pattern. However these changes can only take place if they are consistent with the computers understanding of the picture. Thus although the user may not be aware of it he is in the business of talking about the machines interpretation of the picture. This is acceptable as long as the machines and the users interpretations of the pattern are consistent with each other. However it results in all sorts of problems where the man and the machines interpretations are not consistent or where the mans understanding of the pattern is subject to change. The reason for this is that typically the methods for representing patterns inside a computer are not complete (in the sense of visual stimulus) or flexible enough to handle wholesale change.

The consideration of this problem has led to a different view of computer graphics, figure 59. Here the idea is that both the

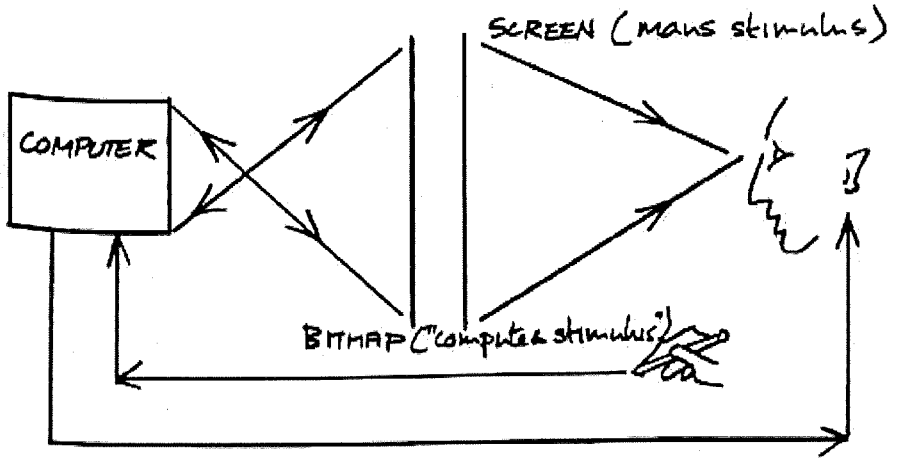


Figure 59. Communicating interpretations.

computer and the man are in receipt of a "stimulus" and that there is a close correspondence between them. In the case of the computer the "stimulus" is a *bitmap* which it has been argued, has a one-to-one correspondence with display points on the screen. The *bitmap*

is, of course, not equivalent to the screen display but it is a good approximation. Just as the man is able to arrive at interpretations of the visual stimulus so the computer must be provided with mechanisms for extracting interpretations from the *bitmap*. Given this the man/machine interaction can be described as a process of communicating interpretations. As the artist decides on actions (that can be a consequence of his interpretation of the screen pattern) he must communicate both the desired action and also (explicitly or otherwise) his interpretation of the visual pattern, or part of the pattern, to which it is applied. Clearly then, the computers interpretative mechanisms should be consistent with the users to a degree that permits his commands to be completed.

It is believed that the shape extraction routines and the facilities for describing figure/ground percepts discussed in chapter 8 indicate both the power and potential of the "communication of interpretations" view of interactive computer graphics and also the extent of the problems to be resolved. It is of interest to catalogue some of them here.

With respect to the "stimulii", one of the problems is to decide how far we need to go in making them equivalent? Visual psychology must have something to offer in resolving this issue. Also there is the question of the interpretive mechanisms. Should we model them on what we know about human psychology or do we need a psychology of computer vision perhaps quite different to mans that nevertheless makes it possible to complete the desired picture manipulations?

Another source of problems is the level and kind of information handled by the mechanisms of interpretation. It has been pointed out previously that some perceived properties of the picture are objectively observable in the "bitmap" whilst other are not. For example a boundary has been described as a physical property. The procedures described in this thesis do not go beyond the detection of boundaries but there are other physical properties that could be considered. Information about tone, colour and shape, amongst others, could be gathered from the *bitmap*. This information could then be put to the use of higher level interpreters (e.g. figure/ground)

dealing with non-physical properties of a picture.

There is also the question of how far we should, or need, to go with picture interpretation. The figure/ground interpreter described in chapter 8 makes no attempt at recognising the objects depicted. Thus no distinction is attempted between figure as horse, dog, tree or whatever. It remains to be seen how useful these higher level interpretive procedures would be.

Considerable work remains to be done both in terms of the interpretive mechanisms and the commands, or actions, the user might wish to realise. In conclusion, the work described in this thesis indicates a different direction for computer graphic systems that are designed for generating and manipulating two dimensional pictures. It is believed that both the usefulness and power of facilities based on procedures that process a *bitmap* (what has been called "the communication of interpretations approach") has been demonstrated and that the way ahead is to examine the issues raised in detail.

REFERENCES

Alexander C., "Notes on the synthesis of form", Oxford Univ Press, Ltd

Barker B. & Cornock S., "ART user guide", Leicester Polytechnic

Bense M., "The projects of generative aesthetics", in Cybernetics, Art & Ideas, ed. Reichardt J., pp 57-60, Studio Vista, 1971.

Bradley D., "Introduction to the Picaso Interpreter, Leicester Polytechnic Computer Centre, 1979.

Cohen, H., "On purpose : an enquiry into possible roles of the computer in art", Studio International, vol. 187, no. 962, pp 9-16, 1974

Crichton M., "Jasper Johns", Thames and Hudson Ltd., 1977.

Cross N., "The automated architect", Pion Books Ltd., 1977.

Csuri C., "Computer graphics and art", Proc. IEEE, vol. 62, no. 4. pp 503-515 , 1974.

Eason K.D., "Dialogue design implications of task allocation between man and computer", Proc. Man-Computer Communication, Loughborough, 43-58, 1979.

Edmonds, E.A., Schappo A. & Scrivener S.A.R., "Graphics without data structures", Proc. C.A.D. 80, pp 138-145, Brighton 1980.

Ernst M., "Beyond Painting, Wittenborn, Schultz Inc., New York 1948.

Evans, C., "Simplifying the user interface : the key to effective man-machine dialogue", Infotech State of the Art Report 1¼, Infotech, 1972.

Foley, J.D. & Wallace V.L., "The art of natural man-machine conversation", Proc. IEEE, vol. 62, no. 4, pp 462-471, 1974.

Franke H.W., "Computer graphics, computer art", Phaidon Press Ltd., London, 1971.

Gaines B.R. & Facey P.V., "Programming interactive dialogues", Computing and People, pp 63-64, Edward Arnold, 1977.

Gilson E., "Painting and reality", Princeton University Press, 1957.

Gombrich E.H., "Leonardo's method for working out compositions", Norm & Form : Studies in the Art of the Renaissance, pp 58-63, Phaidon Press, 1966.

Gregory R.L., "The intelligent eye", Werdenfeld & Nicolson, 1970.

Gregory R.L., "The confounded eye", in : Illusion in Nature and Art, Duckworth, 1973.

Guichard-Meili J., "Matisse", Thames and Hudson, 1967.

Herot C., "Graphical Input through machine recognition of sketches", Computer Graphic, Vd. 10, No II, pp 97-102, 1976.

Julesz B., "Texture and visual perception", Scientific American, vol. 212, no 2, pp 38-48, 1965.

Klee P., "Pedagogical sketchbook", Faber and Faber, 1925.

Knowlton, K.C., "Picture processing by computer", Science, vol. 164, no. 3875, p 19, 1969.

Knowlton, K., "Mini explor : a Fortran coded version of the Explor language for minicomputers", Page 35, Bulletin of the Computer Art Society, 1975.

Mallen G.L., "The cybernetics of design processes", Recent Topics in Cybernetics, University of London, 1974.

McArthur C.D., "LOGO : user guide and reference manual", Bionic Research Reports, no 14, University of Edinburgh, 1973.

McCurdy E., "Leonardo Da Vinci's Note-books", Duckworth Company, 1910.

Mezie L., "Sparta, a procedure irentated programming language for the manipulation of arbitrary line drawings", Information Processing 68, pp 597-604, North Holland, 1969.

Mezie L. & Zivian A., "Arta an interactive animation system", Information Processing 71, North Holland, 1972.

Morellet F., "Text", in catalogue, Arts Council, 1974.

Nake F., "On generative aesthetics - two picture generating programmes", International Symposium on Computer Graphics, sect. 3, vol 2, Brunel University, 1970.

Nash, K. & Williams R.H., "Computer program for srtists : ART 1", Leonardo, vol. 3, pp 439-442, 1970.

Noll M.A., "The digital computer as a creative medium", IEEE Spectrum, vol. 4, no. 10, pp 89-95, 1967.

Negroponte N., "On being creative with computer aided design, Information Processing 77, I.F.I.P., pp 695-704, North Holland, 1977.

Newman W.M. & Sproull R.F., "Principles of interactive computer graphics", McGraw-Hill, 1973.

Newman, W.M., "Raster graphics in CAD", CAD Systems, IFIP, pp 369-377, North Holland, 1978.

O'Shea T., "LOGO", University of Edinburgh, 1974.

Papert, S., "Teaching children thinking", IFIP Conference on Computer Education, North Holland, 1970.

Reichardt J. ed., "Cybernetic serendipity : the computer and the arts", Studio International, London, 1968.

Reichardt J., "The computer in art", Studio Vista, 1971.

Saunders, R. "PLAD and computing for artists", Datafair 73 II, British Computer Society, Nottingham, pp 479-484, 1973.

Scrivener, S.A.R., Edmonds E.A. & Thomas L.A., "Improving image generation and manipulation using raster graphics", Proc. C.A.D. 78, pp 223-229, Brighton, 1978.

Scrivener S.A.R. & Edmonds E.A., "Pictorial properties in raster graphics : classification and use", Proc. Computer Graphics 80, Brighton, 1980.

Scrivener S.A.R. & Schappo A., "GLIMPS : A Graphical Language for the Interactive Manipulation of Perceived Shapes", MCl Report No 38, Leicester Polytechnic, 1981.

Struyken P., "Structure II", Studio International, vo. 185, no. 955 pp 221, 1973.

Sykora Z. & Blazek J., "Computer-aided multi-element geometrical abstract paintings", Leonardo, vol. 3, pp 409-413, 1973.

"Systems catalogue", Arts Council Exhibition, 1972.

Thompson, M., "Computer art : a visual model for the modular pictures of Manuel Barbadillo", Leonardo, vol. 5, pp 219-226, 1972.

Vince, J., "Picaso User Manual", Middlesex Polytechnic.

Williams, R.H., "Statistical shading using digital computer program ART 2", Leonardo, vol. 4, pp 365-375, 1971.

APPENDIX 1 SOME ALGORITHMS

The algorithms in this appendix have been written in such a way that it is hoped they will be comprehendible to the reader without requiring extensive knowledge of computing. To the computer expert it may appear that I have taken liberties. If this is the case I believe it is justified.

MATRICES

The algorithms assume the use of matrices that have the same number and arrangement of locations as the bitmap (conceptually then the bitmap can be regarded as a matrix). Computer memory is assumed to be infinite and this accounts for the multiplicity of matrices. In practice, sparse matrices (1) could be used in the interest of economy of memory. Also redundant matrices could be remployed in some algorithms. To avoid confusion matrices have been cited as needed in the following algorithms irrespective whether some might have been reused

Every location in a matrix can take a value of zero or one. In all matrices other than the bitmap a value of 1 signifies a point of interest and 0 a location outside the range of interest. For example if a region is extracted from the bitmap the *REGION* procedure will produce a matrix in which points equivalent to those in the bitmap have a value of 1 irrespective of the value of the points in bitmap (i.e. it might be the case the extracted region was white in the bitmap). Thus in all matrices generated from the bitmap, either directly or indirectly, a value of 1 means that the point belongs to the region being processed, and a value of 0 that it does not.

The italic capitals represent procedure or process names and where they appear in the definition of a procedure it means that the named procedure would be performed at that point. Every algorithm is also intuitively described. The terms used are defined in Chapter 8.

1 Lindterg E, "An Introduction to Sparse Matrix Techniques", Summer School on Circuit Theory, Prague, Czechoslovakia, September 1974.

INDEX TO ALGORITHMS

REGION ⟨R,I,P⟩

Assumptions

R - a matrix used to hold the REGION extracted from I

I - a matrix from which the REGION is to be extracted

P - a point in the REGION e.g. its location and insensity

Description

The point P used to identify the REGION of interest and can be regarded as a location (giving its position) and a value (giving its intensity). In the following algorithms P will be referred to as "location P" or "value P".

A location P is provided which identifies the area of interest. Given this location a REGION can be extracted from I and stored in R.

Process

```
BEGIN
    Set the value of location P in matrix R to 1
    Put location P on the stack (S)
    WHILE S is not empty
        DO NEIGHBOUR1 ⟨R,I,P⟩
END
```

NEIGHBOUR1 ⟨A,B,P⟩

Assumptions

A - a matrix used to hold the REGION extracted from B

B - the matrix from which the REGION is to be extracted

P - a point in matrix B

Description

A REGION is extracted from B. This is done by iteratively considering points in B, processing their next neighbour points if they haven't already been processed and have the same intensity as the value of P. Processing a next neighbour involves setting the equivalent point in A to 1 and entering it onto a stack. Processing ceases when the stack is empty. The stack begins to empty as next neighbours are found to be processed or are not of the correct intensity.

Process

```
BEGIN
    Take the top point (TP) off S
    UNTIL all next neighbours of TP have been considered
          DO if next neighbour (NN) has a value of 0 in matrix A
             THEN if NN in B has a value of value P
                  THEN set matrix A, location NN to 1
                       and put location NN onto S.
END
```

COPY ⟨A,B,P⟩

Assumptions

A - the matrix into which a REGION is to be COPIED

B - the matrix containing the REGION to be COPIED into A

P - a point in the REGION to be COPIED.

Description

Points with a value of P in B are COPIED with a value of P into the equivalent locations in A.

Process

```
BEGIN
    UNTIL all locations in the matrices have been considered
        DO if location (L) in matrix B has a value P
        THEN put the value of P in L of matrix A
END
```

FILL ⟨R,P⟩

Assumptions

R - a matrix containing the REGION to be FILLED

P - a point in the REGION to be FILLED

Description

First the REGION to be FILLED is extracted. Then the value of P is inverted since to FILL a REGION is to replace all the points in the REGION by the inverse or complement of their value. The points in F are then COPIED into the original matrix by setting these points to the value of IP.

Process

```
BEGIN
    REGION ⟨F,R,P⟩
    Let IP be the inverse of the value of P in R
    COPY ⟨R,F,IP⟩
END
```

FINDPOINT ⟨P,A⟩

Assumptions

A - a matrix

P - a point in the matrix A i.e. its location and intensity

Description

This procedure scans through the matrix A looking for a point with a value of 1. As soon as this condition is met P is set to the location of the point that meets the condition.

Process

```
BEGIN
    UNTIL current point (CP) in matrix A has a value of 1
          OR CP is the last point in A
          DO make the next point to be considered in A the current point
    Set P to the current point
END
```

REGIONPLUS ⟨RP,R,P⟩

Assumptions

RP - a matrix used to hold the REGION extracted from R

R - a matrix used to hold the REGION of interest, and only the REGION of interest to be extracted

P - a point in the REGION of interest

Description

A REGION is extracted that includes the COMPLEMENTARY BOUNDARIES of the REGION in R.

Process

BEGIN

    Set the value of location P in matrix RP to 1

    Put location P on the stack(S)

    WHILE S is not empty

        DO *NEIGHBOUR1* ⟨RP,R,P⟩

END

NEIGHBOUR2 ⟨A,B,P⟩

Assumptions

A - a matrix used to hold the REGION to be extracted from B

B - a matrix from which the REGION is to be extracted

P - a point in matrix B

Description

When the REGION is being extracted all neighbouring points of a complementary value to the points in the REGION being processed are also extracted, Thus the external boundaries of the REGION of interest in B are included in A

Process

```
BEGIN
     Take the top point (TP) off S
     UNTIL all next neighbours of TP have been considered
           DO if next neighbour (NN) has a value of 0 in matrix A
              THEN if NN in B has a value P
                   THEN set matrix A, location NN to 1
                        and put location NN onto S
                        and set all complementary neighbours
                        locations of NN in A to 1
END
```

DIFFERENCE ⟨A,B,C⟩

Assumptions

A - a matrix into which the points of B that are different to C are placed

B - a matrix

C - a matrix

Description

Equivalent points in matrix B and C are considered for the whole of the matrix and the value of the point in C is subtracted from the value of the point in B.

Process

```
BEGIN
     UNTIL all points in matrix B have been considered
     DO if the value of location (P) in B has a value 1
          THEN set matrix A, location P to the value of the value
               of matrix B, location P  minus the value of matric C,
               location P
          ELSE set the value of matrix A, location P to 0
END
```

COMPLEMENTARYBOUNDARIES ⟨CB,R,P⟩

Assumptions

CB - a matrix used to hold the COMPLEMENTARY BOUNDARIES of the REGION of interest in R

R - a matrix containing the REGION of interest

P - a point in the REGION of interest

Description

Given a REGION the REGION plus its COMPLEMENTARY BOUNDARIES are generated. The DIFFERENCE between the REGION plus complementary boundaries and the original REGION yields the COMPLEMENTARY BOUNDARIES

Process

```
BEGIN
    REGIONPLUS ⟨RP,R,P⟩
    DIFFERENCE ⟨CB,RP,R⟩
END
```

OUTERCOMPLEMENTARYBOUNDARY ⟨OCB,R,P⟩

Assumptions

OCB - a matrix used to hold the OUTER COMPLEMENTARY BOUNDARY from matrix R

R - the matrix from which OCB is to be extracted

P - a point in the REGION of interest

Description

Given a matrix containing a REGION the COMPLEMENTARY BOUNDARIES are derived. A point is the located on the OUTER COMPLEMENTARY BOUNDARY. Since boundaries are 'regions' the OUTER COMPLEMENTARY BOUNDARY can be extracted as a REGION from CB

Process

BEGIN

COMPLEMENTARYBOUNDARIES ⟨CB,R,P⟩

FINDPOINT ⟨L,CB⟩

REGION ⟨OCB,CB,L⟩

END

FIGURE ⟨F,I,P⟩

Assumptions

F - a matrix used to hold the FIGURE extracted from matrix I

I - the matrix from which the FIGURE is to be extracted

P - a point in FIGURE

Description

A FIGURE is equivalent to a REGION and consequently it is simply a matter of extracting a region

Process

BEGIN

REGION ⟨F,I,P⟩

END

GROUND ⟨G,I,P⟩

Assumptions

G - a matrix used to hold the GROUND extracted from matrix I

I - the matrix from which the GROUND is to be extracted

P - a point in the GROUND

Description

A REGION is extracted from I representing the visible GROUND. The OUTER COMPLEMENTARY BOUNDARY of this REGION is derived. The REGION enclosed within the OUTER COMPLEMENTARY BOUNDARY is the GROUND and this is extracted

Process

```
BEGIN
    REGION ⟨R,I,P⟩
    Set location IP to location P
    Set value of IP to 1
    OUTERCOMPLEMENTARYBOUNDARY ⟨OCB,R,IP⟩
    Set value of IP to 0
    REGION ⟨G,OCB,IP⟩
END
```

FIGUREANDGROUND ⟨F,G,I,P⟩

Assumptions

F - matrix used to hold FIGURE extracted from matrix I

G - matrix used to hold GROUND extracted from matrix I

I - the matrix from FIGURE and GROUND are to be extracted

P - a point in the FIGURE

Description

A FIGURE is extracted and a point is located on its OUTER COMPLEMENTARY BOUNDARY. This point belongs to the GROUND and can be used to extract the GROUND

Process

BEGIN

FIGURE ⟨F,I,P⟩

Set location IP to location P

Set value IP to 1

OUTERCOMPLEMENTARYBOUNDARY ⟨OCB,F,IP⟩

FINDPOINT ⟨FP,OCB⟩

Set value FP to the complement of value P

GROUND ⟨G,I,FP⟩

END

FIGURESANDGROUND ⟨FS,G,I,P⟩

Assumptions

FS - matrix used to hold the FIGURES extracted from matrix I

G - matrix used to hold the GROUND extracted from matrix I

I - matrix from which FIGURES and GROUND is to be extracted

P - a point in the GROUND

Description

The GROUND is extracted. The "visible" GROUND is then extracted using REGION. The DIFFERENCE between the GROUND and the "visible" GROUND yields the FIGURES

Process

BEGIN

GROUND ⟨G,I,P⟩

REGION ⟨R,I,P⟩

DIFFERENCE ⟨FS,G,R⟩

END

FIGURESONGROUND ⟨FS,I,P⟩

Assumptions

FS - a matrix used to hold the FIGURES extracted from I

I - matrix from which FIGURES are derived

P - a point in the GROUND

Description

This is essentially the same process as FIGURESANDGROUND. The GROUND is discarded.

Process

```
BEGIN
    FIGURESANDGROUND ⟨FS,G,I,P⟩
END
```

GROUNDUNDERFIGURE ⟨G,I,P⟩

Assumptions

G - a matrix used to hold the GROUND extracted from I

I - matrix from which GROUND is derived

P - a point in the FIGURE

Description

This is essentially the same process as FIGUREANDGROUND. The FIGURE is discarded.

Process

```
BEGIN
    FIGUREANDGROUND ⟨F,G,I,P⟩
END
```

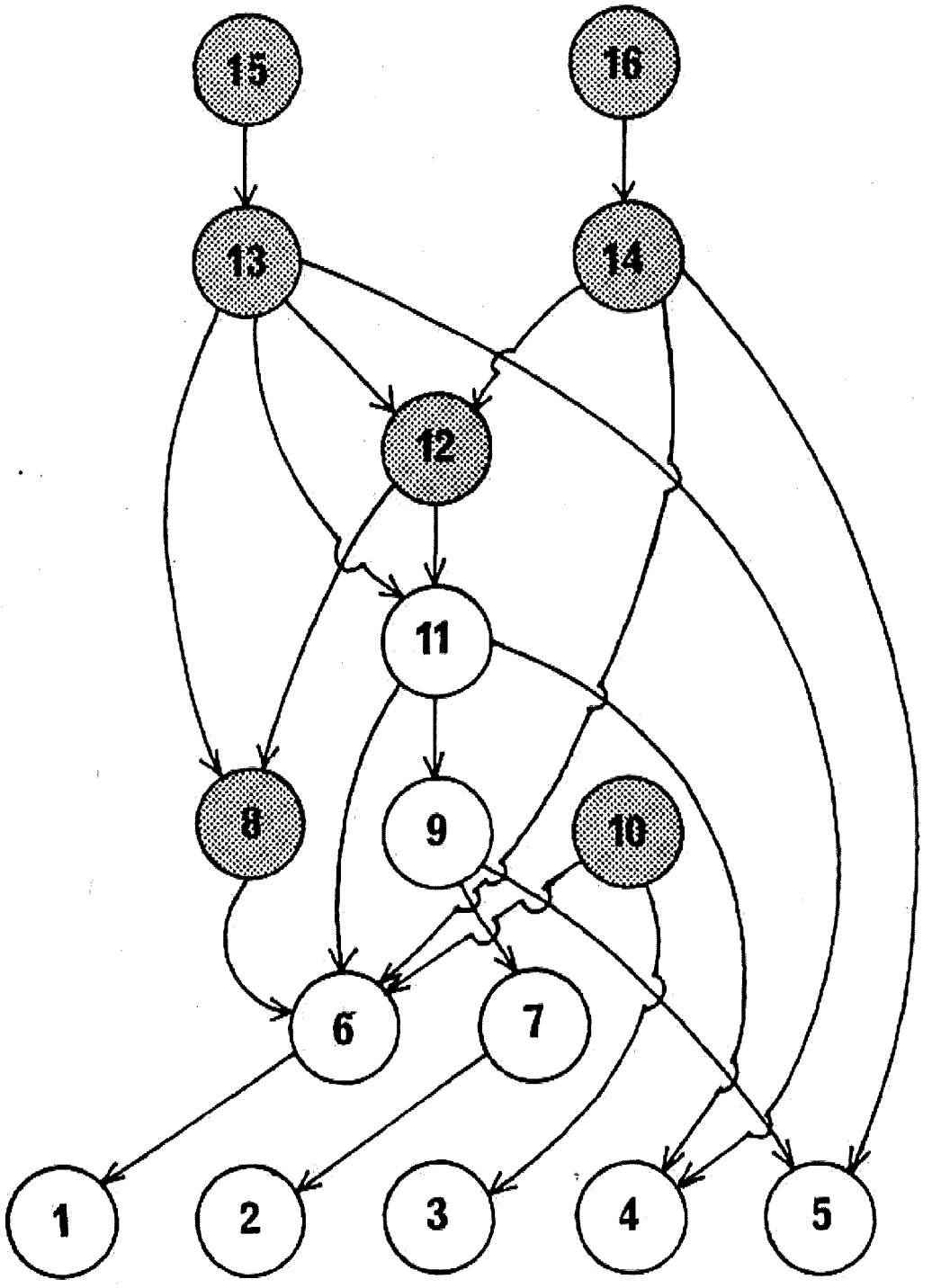
LEV
LEV
LEV
LEV
LEV
LEV
LEV
Tat
mo
ar
ot
bu

TABLE 1

LEVEL 7	GROUNDUNDERFIGURE 15		FIGURESONGROUND 16		
LEVEL 6	FIGUREANDGROUND 13		FIGURESANDGROUND 14		
LEVEL 5	GROUND 12				
LEVEL 4	OUTERCOMPLEMENTARYBOUNDARY 11				
LEVEL 3	FIGURE 8	COMPLEMENTARYBOUNDARIES 9		FILL 10	
LEVEL 2	REGION 6		REGIONPLUS 7		
LEVEL 1	NEIGHBOUR1 1	NEIGHBOUR2 2	COPY 3	FINDPOINT 4	DIFFERENCE 5

Table 1 illustrates the heirarchy of function. Level 1 functions are the most basic and do not make use of any other functions. Level 7 functions are the least basic and make use of lower level functions. Functions at other levels (e.g. 4) make use of lower level functions (e.g. level 4 or 2) but not those at higher levels

FUNCTION HEIRARCHY



Functions available in GLIMPS

Functions not available in GLIMPS (however these could be made available to GLIMPS if necessary)

APPENDIX 2 LOGO

LOGO - BASIC CONCEPTS

LOGO is an interpretive language where a command is entered at a teletype, or other input device. The execution of a command involves, without exception the evaluation of a function. Once a command has been executed a prompt is output to indicate readiness for the next command. This sequence is repeated until terminated by explicit command from the user. Every command consists of a function name specifying the action to be taken and zero or more arguments. Evaluation of a command, without exception produces a single result ; in addition to a result evaluation of some functions also produces an effect, for example, a line is drawn on a display. The result from a function evaluation is an actual piece of LOGO data which is held in readiness in a particular place in LOGO's workspace. The result of a function evaluation only remains available during the execution of a single command but the final result of a command is assigned to a unique location in LOGO's workspace whence it can be retrieved as the result of evaluating a special function, which has no arguments. It follows from above that a typical command may involve the evaluation of a number of nested functions. C.D. McArthur has described the execution of a command in the following way, "it is convenient to think of the input and execution of a command as the effect produced by the evaluation of an unnamed primeval (sic) function of one argument. Execution proceeds simply because the evaluation of that function is outstanding, and continues until its argument is available. The argument is, of course, the final value left over from the users command, and when this has been assigned to its special location the evaluation of primeval function is complete ; i.e. the command has been executed".

LOGO contains many built in functions, some of which are used for their effect and others for their result. These built in functions provide a base language which can be used, more-or-less, directly by the user. Functions are also provided which the user may employ to define functions. User defined functions are a numbered sequence of commands with associated name ; commands not being executed while the definition is in progress ; they are stored as text to be retrieved and executed when a use of the function name calls for its execution. Once defined the user defined

function (U.D.F.) can be used in the same way as the built in functions. U.D.F.'s may have arguments which appear in the definition as names which are used to refer to unknown values.

The U.D.F. is a powerful facility in the LOGO language rather like sub-routines in other languages but it has the advantage of appearing, in use, as any other function and consequently appears as a way of extending, or personalising, the language.

The language supports two data types ; words and lists. Words which consist solely of numeric characters are called numeric words and are treated as values. A list is an ordered sequence of elements where an element can be a word or a list.

SYNTAX - ELEMENTS OF THE LOGO LANGUAGE

The basic unit of the language is a character group. Depending on special delimiters a character group in a command is interpreted by LOGO as one of the following.

1. a word (a piece of actual data)
2. the name of a piece of data
3. the name of a function
4. an element of a list.

The difference between a word as stored by LOGO needs some further clarification. A word described as a string of characters is reasonably descriptive of how LOGO stores it. When the user wishes to create a name to refer to a value a word and the value are handed over as arguments to a function which creates a name out of the characters in the word and stores it with the value as an associated pair. However the name is in a different form to the word used to create it and cannot be manipulated.

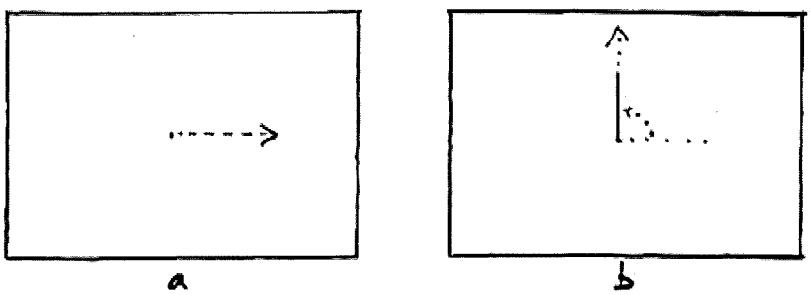
Some versions of LOGO allow a user function to manipulate the text comprising a function definition, including its own ; therebye dynamically altering the functions behaviour. A full definition of the syntax of LOGO is not necessary here, and in any case depends to some extent on the implementation but the syntax of the Edinburgh University implementation of LOGO can be found in reference

WHAT LOGO LOOKS LIKE - SOME EXAMPLES

The flavour of LOGO can be illustrated by a number of examples based on the version of the language implemented at Edinburgh which involves graphical functions ( ). This system allows the generation of pictures on a number of drawing devices, a turtle, a visual display and plotter. It therefore includes functions for selecting and control of devices. The drawing mechanism employed in LOGO is unconventional ; the drawing device is considered to have an orientation and a current position. The commands provided allow the user to modify the device orientation or move it forwards or backwards n units in its current orientation. For example if the pen was at the position and orientation illustrated in figure 1 a, and the user issued the commands :-

```
1 : LEFT 90
1 : FORWARD 50
1 :
```

figure 1 b would result.



Orientation function arguments are assumed by LOGO to be degrees whilst the precise meaning of a unit of distance is dependent upon the particular device being used. The 1 : prefixing commands is the prompt used by LOGO to indicate readiness for the next command.

Thus to generate a square of side 50 units the user could type :-

```
1 :  FORWARD 50
1 :  LEFT 90
1 :  FORWARD 50
1 :  LEFT 90
1 :  FORWARD 50
1 :  LEFT 90
1 :  FORWARD 50
1 :  LEFT 90
1 :
```

This could be defined as a user defined function and then executed by reference to its name

```
1 :  TO SQUARE
8 :  1Ø FORWARD 5Ø
8 :  2Ø LEFT 9Ø
8 :  3Ø FORWARD 5Ø
8 :  4Ø LEFT 9Ø
8 :  5Ø FORWARD 5Ø
8 :  6Ø LEFT 9Ø
8 :  7Ø FORWARD 5Ø
8 :  8Ø LEFT 9Ø
8 :  END
1 :  SQUARE
```

This can be made more modular by first defining the side instructions as a U.D.F.. Thus :-

```
1 :  TO SIDE
8 :  1Ø FORWARD 5Ø
8 :  2Ø LEFT 9Ø
8 :  END
1 :  TO SQUARE
8 :  1Ø SIDE
8 :  2Ø SIDE
8 :  3Ø SIDE
8 :  4Ø SIDE
8 :  END
1 :  SQUARE
```

```
1 : TO SIDE LENGTH
8 : 1Ø FORWARD LENGTH
8 : 2Ø LEFT 9Ø
8 : END
1 : TO SQUARE LENGTH
8 : REPEAT 4 SIDE
8 : END
1 : SQUARE 1ØØ
1 : SQUARE 32
```

The right hand example demonstrates how the user defined function can be made more general by the inclusion of a parameter LENGTH in the definition of the function.

APPENDIX 3 LOGO SESSION

```
WHO ARE YOU : TIM
TIM IS ALREADY LOGGED ON
WHO ARE YOU : CALUM

LOGO READY
1: GETDISPLAY
YOU HAVE BEEN CONNECTED
1: FORWARD 100
1: LEFT100
LEFT100 IS NOT A WORD I UNDERSTAND
1: LEFT 100
1: LEFT 100
1: LEFT 100
1: CLEAR
1: FORWARD 100
1: LEFT 100
1: CLEAR
1: FORWARD 100
1: LEFT
LEFT NEEDS 1 MORE INPUTS
1: LEFT 150
1: LEFT 100
1: LEFT 100
1: END
END IS NOT A WORD I UNDERSTAND
1: CLEAR
1: FORWARD 10
1: FORWARD 234
1: BACKWARDS348
BACKWARD NEEDS 1 MORE INPUTS
1: BACKWARD 348
1: CLEAR
1: RIGHT 77
1: LEFT244
LEFT244 IS NOT A WORD I UNDERSTAND
1: LEFT 244
1: RIGHT 5
1: TO SQ
&: 1 FORWARD 100
&: 2 LEFT 100
&: 2 LEFT 100
&: 3 LEFT 100
&: END
SQ DEFINED
1: SQ
1: TO SQ
SQ REDEFINED
&: 1 FORWARD 100
&: 2 LEFT 100
&: 3 BACKWARD 100
&: 4 RIGHT 100
&: END
SQ DEFINED
1: SQ
1: CLEAR
1: SHOW SQ
```

```
TO SQ
 1 FORWARD  100
 2 LEFT  100
 3 BACKWARD  100
 4 RIGHT  100
END


1: F
F NEEDS 1 MORE INPUTS
1: CHANGE
SOMETHING MISSING
1: CHANGE SQ
&: 1 FORWARD 100 LEFT 90
&: 2 FORWARD 100 LEFT 90
&: 3 FORWARD 100 LEFT 90
&: 4 FORWARD 100
&: END
SQ DEFINED
1: SQ
LEFT  90 IS EXTRA. I IGNORED IT
IN SQ
 1 FORWARD  100 LEFT  90
1: CHANGE
SOMETHING MISSING
1: CHANGE SQ
&: 1 FORWARD 100
&: LEFT 90
YOU FORGOT THE LINE NUMBER
&: 2 LEFT 90
&: 3 FORWARD 100
&: 4 LEFT 90
&: 5 FORWARD 100
&: 6 LEFT 90
&: 7 FORWARD
&: END
SQ DEFINED
1: CLEAR
1: SQ
FORWARD NEEDS 1 MORE INPUTS
IN SQ
 7 FORWARD
1: CHANGE SQ
&: 7 FR
&: 7 FORWARD 100
&: SHOW


TO SQ
 1 FORWARD  100
 2 LEFT  90
 3 FORWARD  100
 4 LEFT  90
 5 FORWARD  100
 6 LEFT  90
 7 FORWARD  100
END
```

```
&: CLEAR
YOU FORGOT THE LINE NUMBER
&: EXIT
YOU FORGOT THE LINE NUMBER
&: END
SQ DEFINED
1: CLEAR
1: SQ
1: LEFT 45
1: FORWARD 120
1: LEFT 45
1: FORWARD100
FORWARD1 IS NOT A WORD I UNDERSTAND
1: FORWARD 100
1: LEFT 135
1: FORWARD 120
1: RIGHT 45
1: FORWARD 100
1: RIGHT 135
1: FORWARD 120
1: RIGHT 45
1: FORWARD 100
1: CLEAR
```

USING GLIMPS: AN EXAMPLE

```
ENABLE PEN
DOWN
GO 116 86
GO 184 86
GO 184 159
GO 116 159
GO 116 86
UP
GO 125 98
DOWN
GO 171 98
GO 171 148
GO 125 148
GO 125 98
UP
GO 132 105
DOWN
GO 138 105
GO 138 116
GO 145 116
GO 145 123
GO 132 123
GO 132 105
UP
GO 149 123
DOWN
GO 166 123
GO 166 130
GO 161 130
GO 161 143
GO 155 143
GO 155 130
GO 149 130
GO 149 123
```

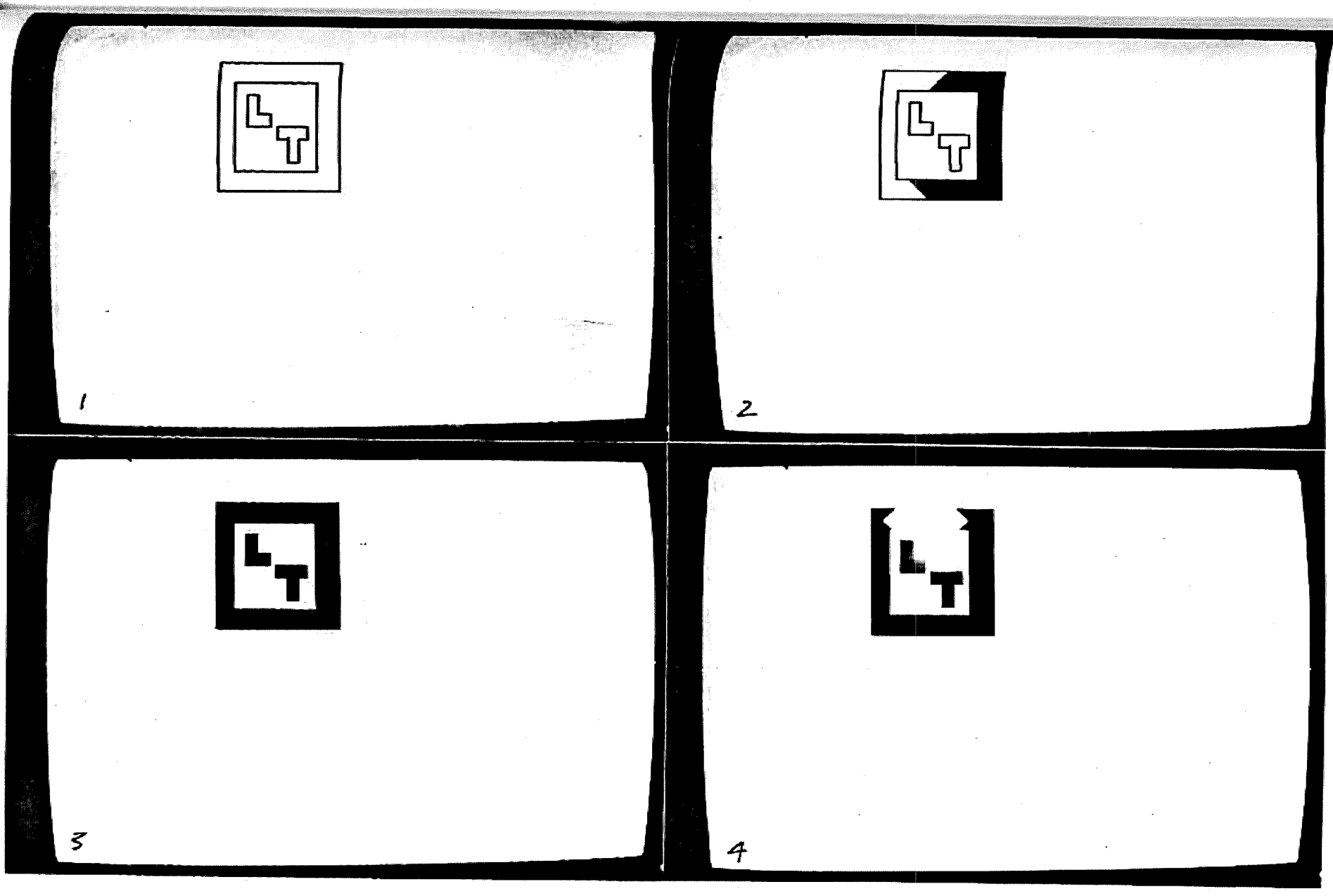
```
TO FILL 117 87
TO FILL 133 106
TO FILL 150 124
TO COPY-FIGURE
10 FIGURE 184 86
20 DISPLAY FIGURESHAPE 281 179
DONE
COPY-FIGURE
TO MOVE-FIGURE
10 FILL 281 179
20 DISPLAY FIGURESHAPE 281 88
DONE
MOVE-FIGURE
UP
GO 119 197
DOWN
GO 214 197
GO 214 264
GO 119 264
GO 119 197
FILL 119 197
TO COPY-GROUND
10 GROUND 281 88
20 DISPLAY GROUNDSHAPE 295 184
DONE
COPY-GROUND
TO COPY-FIGURES-ON-GROUND
10 FIGURES-ON-GROUND 166 123
20 DISPLAY FIGURESHAPE 252 115
DONE
COPY-FIGURES-ON-GROUND
```

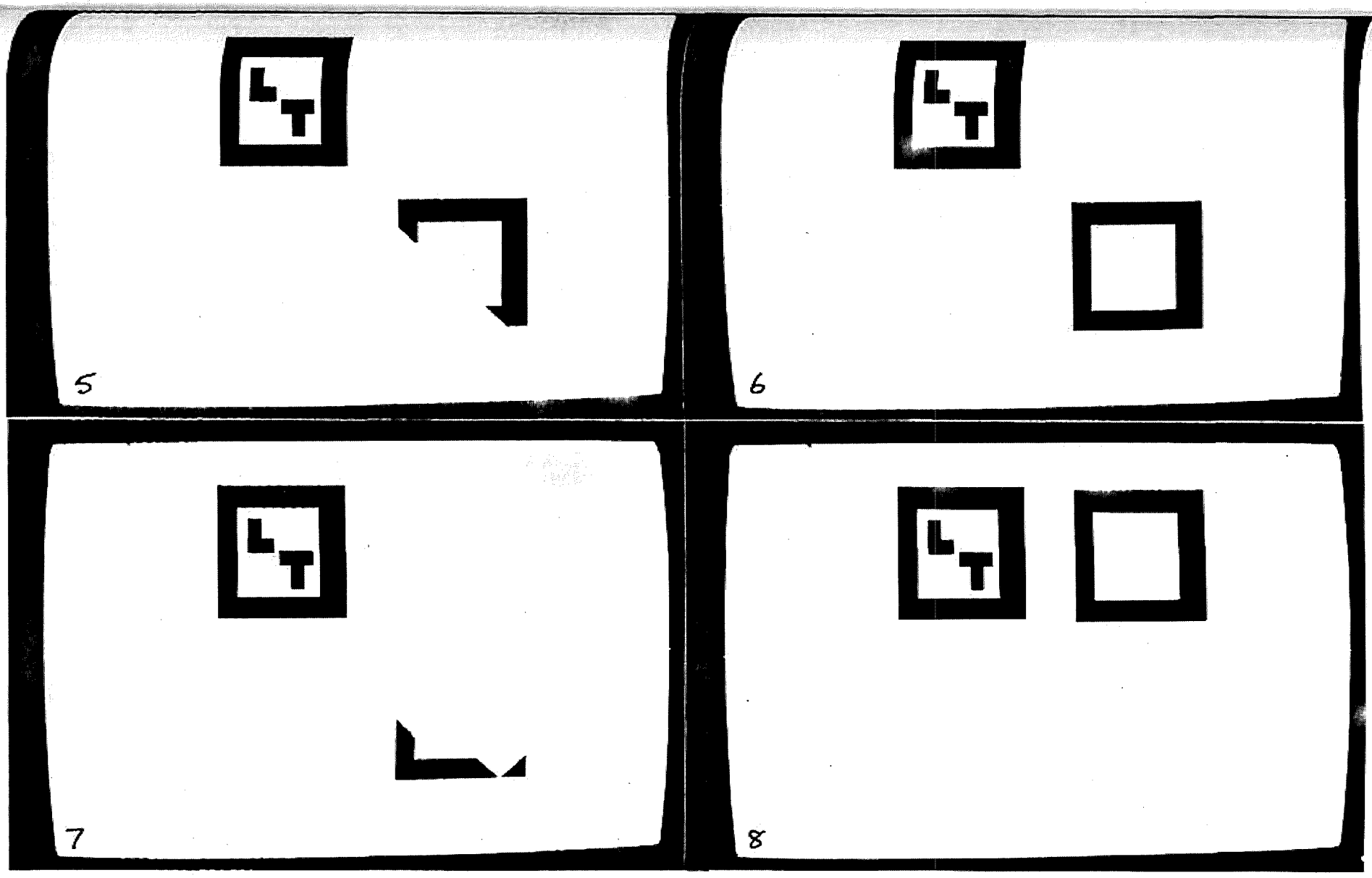
```
TO MOVE-FIGURES-AND-GROUND
10 FIGURES-AND-GROUND 281 88
20 DISPLAY GROUNDSHAPE 255 153
30 DISPLAY FIGURESHAPE 255 153
DONE
MOVE-FIGURES-AND-GROUND
```

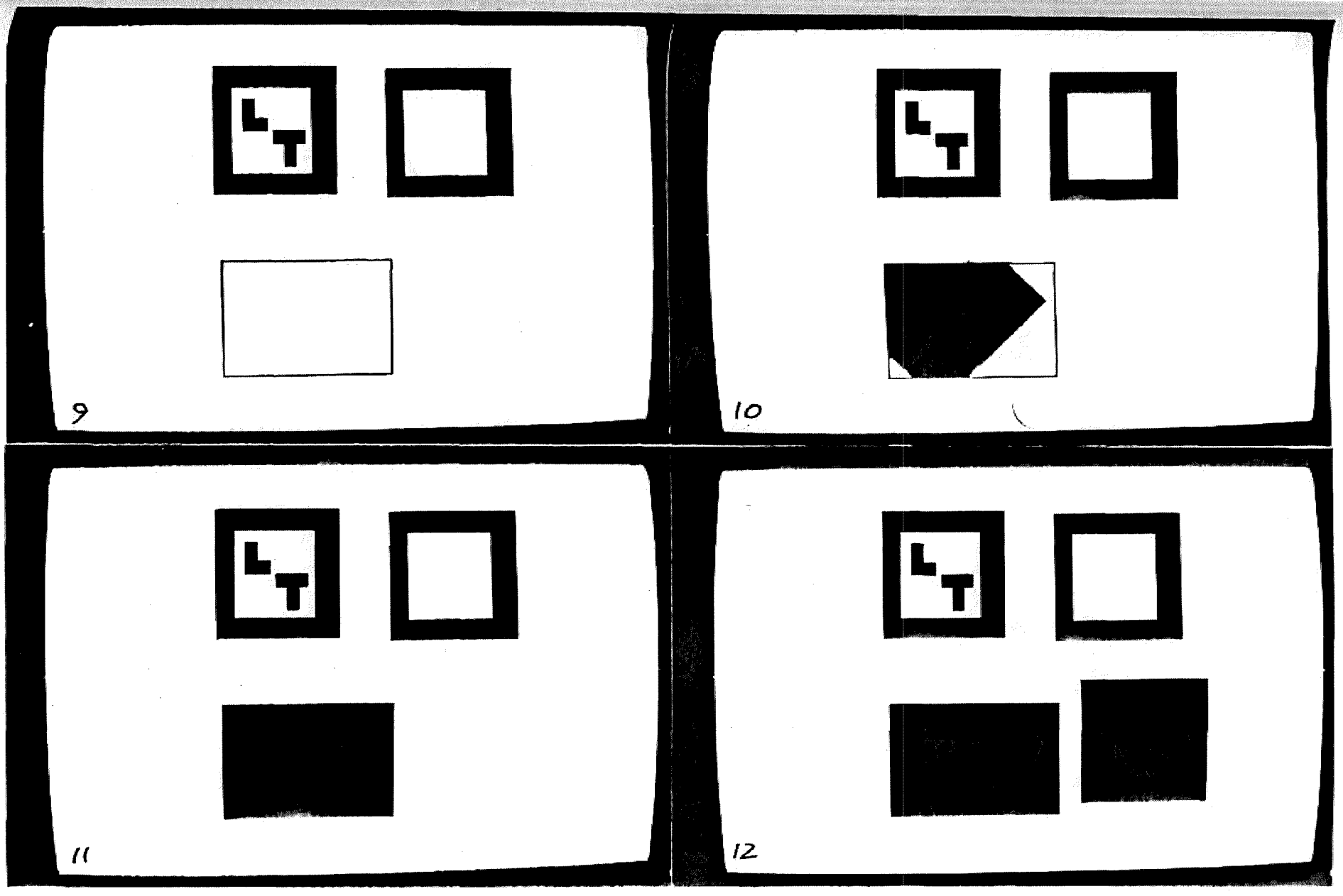
14

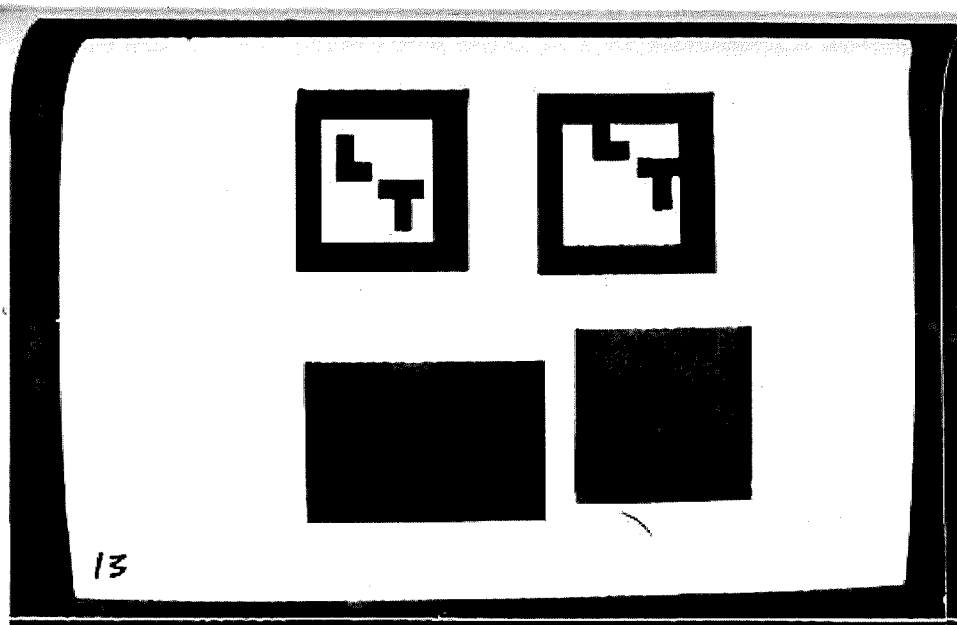
Default names are assigned to shapes derived by extraction routines.

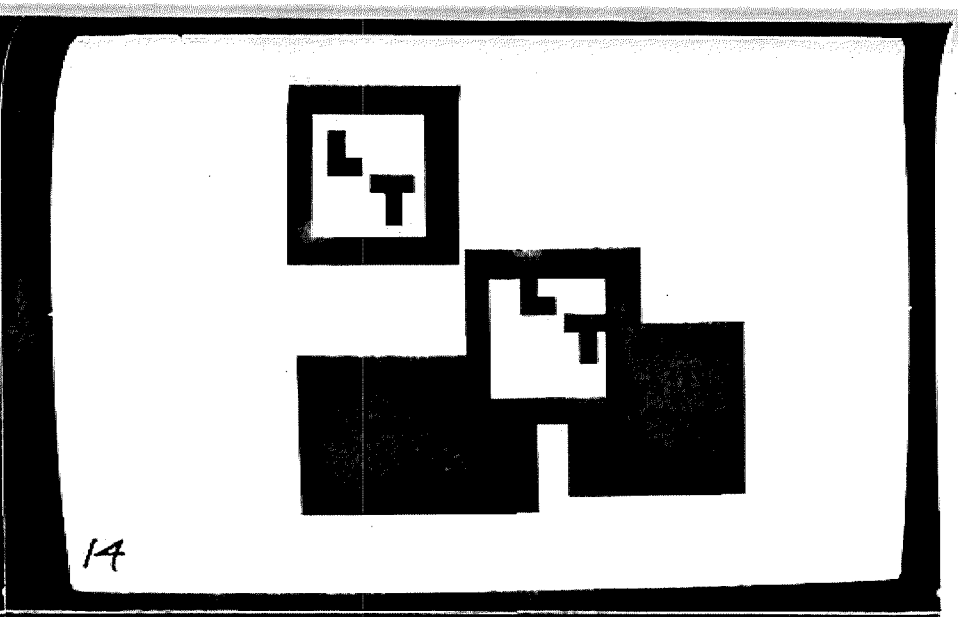
Figure(s) are saved under the name FIGURESHAPE
Ground is saved under the name GROUNDSHAPE

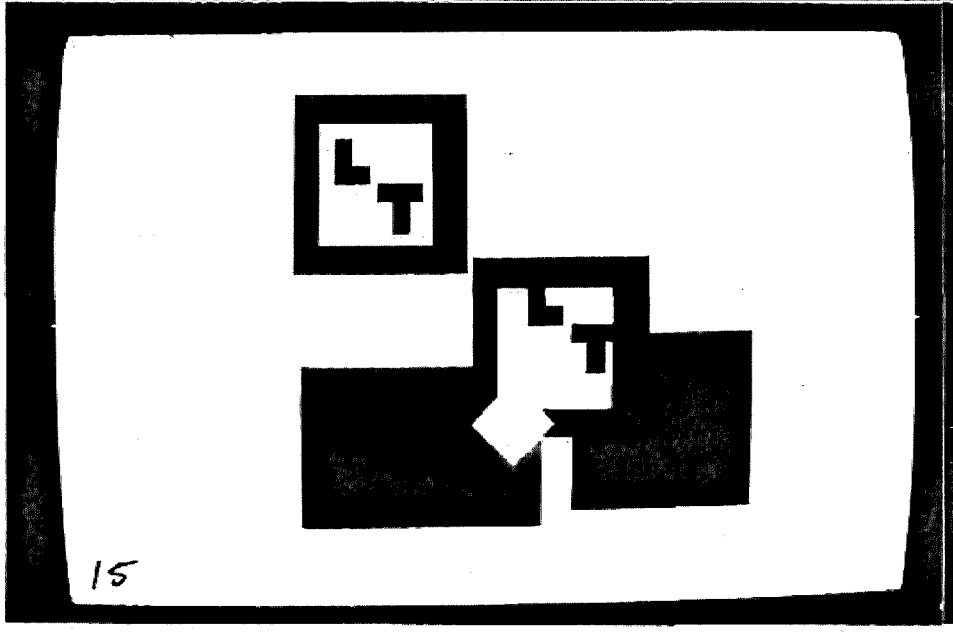

1
2
3
4

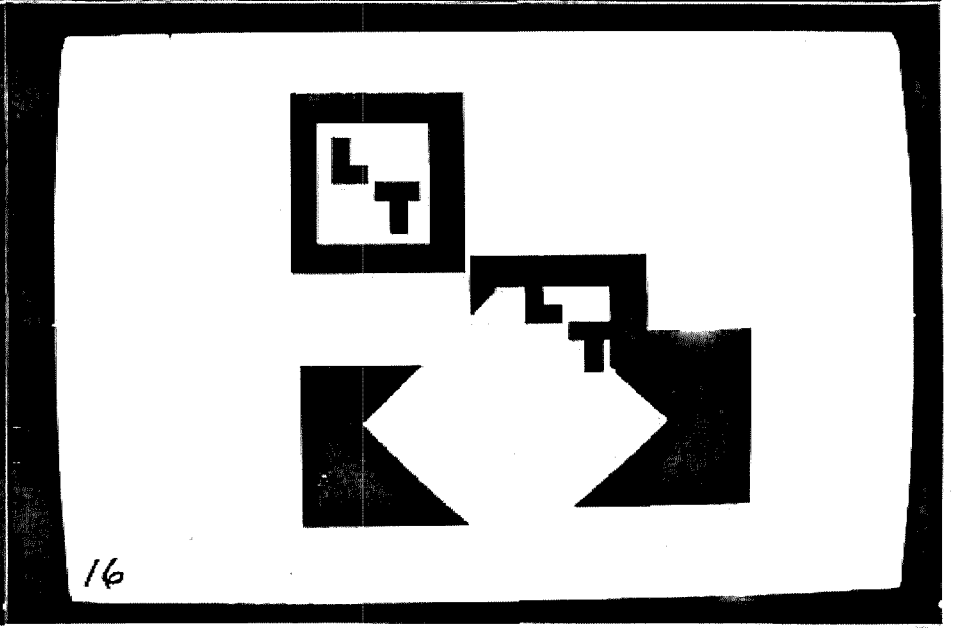

5
6
7
8


9
10
11
12


13


14


15


16